StarFive 赛昉科技

VisionFive 2 Single Board Computer Software Technical Reference Manual

Version: 1.31

Date: 2024/07/01

Doc ID: VisionFive2-TRMEN-001

Legal Statements

Important legal notice before reading this documentation.

PROPRIETARY NOTICE

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to the product development. Guangdong StarFive Technology Co., Ltd.(hereinafter "StarFive") reserves the right to make changes without further notice to any products herein.

StarFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

StarFive does not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

All material appearing in this document is protected by copyright and is the property of StarFive. You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. StarFiveauthorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services.

Contact Us

Address: Room S201, Zone A, No. 2, Haoyang Road, Yunlu Community, Daliang Subdistrict, Shunde District, Foshan, Guangdong, China, 528300

Website: http://www.starfivetech.com http://www.starfivetech.com

Email: sales@starfivetech.com(sales) , support@starfivetech.com(support)

Preface

About this guide and technical support information.

About this document

This document mainly describes how to compile firmware, U-Boot, Linux Kernel and make file systems.

Note:
StarFivehas provide developers with 2 kinds of document versions: Web and PDF. When executing commands, please copy commands in the Web page to avoid errors.

Version History

Table 0-1 Version History

Version	Released	Revision
1.31	2024/07/01	• Added a step in Software Environment *(on page 16)* and Compile Kernel, Device Tree, and Driver Module *(on page 30)*. • Added a tip in Update Configuration Files *(on page 34)*. • Revised minor errors of the command in Creating SPL File *(on page 12)*, Compiling OpenSBI *(on page 14)* and Creating fw_payload File *(on page 15)*.
1.3	2024/05/11	Added the following parts: • Compile and Update Linux Kernel *(on page 16)*. • Replace the dtb Files to be Loaded *(on page 36)*. • Appendix *(on page 53)*
1.2	2023/05/17	Updated steps in Software Environment *(on page 16)*.
1.1	2023/03/03	Updated steps in Creating SPL File *(on page 12)*.
1.0	2022/12/26	The first official release.

Notes and notices

The following notes and notices might appear in this guide:

- **Tip:**
  Suggests how to apply the information in a topic or step.

- **Note:**
  Explains a special case or expands on an important point.

- **Important:**
  Points out critical information concerning a topic or step.

- **CAUTION:**
  Indicates that an action or step can cause loss of data, security problems, or performance issues.

- **Warning:**
  Indicates that an action or step can result in physical harm or cause damage to hardware.

Contents

List of Tables

List of Figures

1. Required Hardware

Make sure that the following hardware are prepared for the operation described in this manual:

- VisionFive 2
- Micro SD card (32 GB or more)
- USB card reader for your host PC
- PC with Linux/Windows/Mac OS
- Power adapter
- USB Type-C Cable
- For desktop environment usage:
  - Keyboard and mouse
  - Monitor or TV
  - HDMI cable
- Additionally, here are some optional components which you may also need:
  - Ethernet LAN cable or a compatible WiFi dongle (ESWIN6600U or AIC8800 module is enabled by default)
  - USB to UART Serial converter module

Tip:
This is used for system recovery via UART boot mode.

Note:
In this guide, Ubuntu 18.04 LTS is installed on the host PC.

2. Making General System

This chapter describes how to make a general system.

It contains the following sections:

2.1. Compiling U-boot and Kernel

This chapter describes how to compile the U-Boot and kernel.

It contains the following sections:

2.1.1. Set Up Compilation Environment

You can follow the steps below to set up your cross-compile.

1. Execute the following commands to install the `riscv64-linux-gnu-gcc` compiler from Ubuntu packages.

   ```
   sudo apt update
   sudo apt upgrade
   sudo apt install gcc-riscv64-linux-gnu
   ```

2. Execute the following command to check the version of the `riscv64-linux-gnu-gcc`.

   ```
   riscv64-linux-gnu-gcc -v
   ```

   The output will be as follows:

Result:

Figure 2-1 Example Output

```
ryan@ubuntu:~$ riscv64-linux-gnu-gcc -v
Using built-in specs.
COLLECT_GCC=riscv64-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc-cross/riscv64-linux-gnu/7/lto-wrapper
Target: riscv64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 7.5.0-3ubuntu1~18.04' --with-
bugurl=file:///usr/share/doc/gcc-7/README.Bugs --enable-languages=c,c++,d,fortran,objc,obj-c
++ --prefix=/usr --with-gcc-major-version-only --program-suffix=-7 --enable-shared --enable-
linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --li
bdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstdcxx-debug --
enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disa
ble-libitm --disable-libsanitizer --disable-libquadmath --disable-libquadmath-support --enab
le-plugin --with-system-zlib --enable-multiarch --disable-werror --disable-multilib --with-a
rch=rv64imafdc --with-abi=lp64d --enable-checking=release --build=x86_64-linux-gnu --host=x8
6_64-linux-gnu --target=riscv64-linux-gnu --program-prefix=riscv64-linux-gnu- --includedir=/
usr/riscv64-linux-gnu/include
Thread model: posix
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
```

2.1.2. Compiling the U-Boot

Follow the steps below to compile the U-Boot for VisionFive 2.

1. Locate to your desired directory to store the U-Boot files. For example, the home directory.
   Example:

   ```
   cd ~ # home directory
   ```

2. Download the source code for U-Boot compilation.

   ```
   git clone https://github.com/starfive-tech/u-boot.git
   ```

3. Switch to the code branch by executing the following command:

   ```
   cd u-boot
   git checkout -b JH7110_VisionFive2_devel
    origin/JH7110_VisionFive2_devel
   git pull
   ```

4. Type the following to compile U-Boot under the U-Boot directory.

   ```
   make <Configuration_File> ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
   make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
   ```

   Tip:

   Configuration_File: For VisionFive 2, the file is `starfive_visionfive2_defconfig`.

Result:

There will be these 3 files generated after compilation inside the `u-boot` directory:

- `u-boot.bin`
- `arch/riscv/dts/starfive_visionfive2.dtb`
- `spl/u-boot-spl.bin`

Figure 2-2 Example Output - u-boot.bin

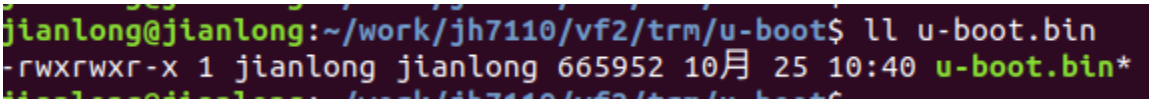


Figure 2-3 Example Output - visionfive2.dtb

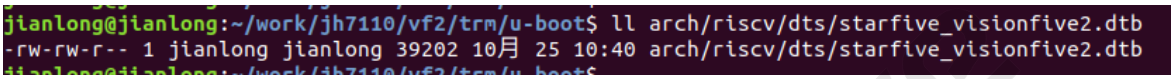
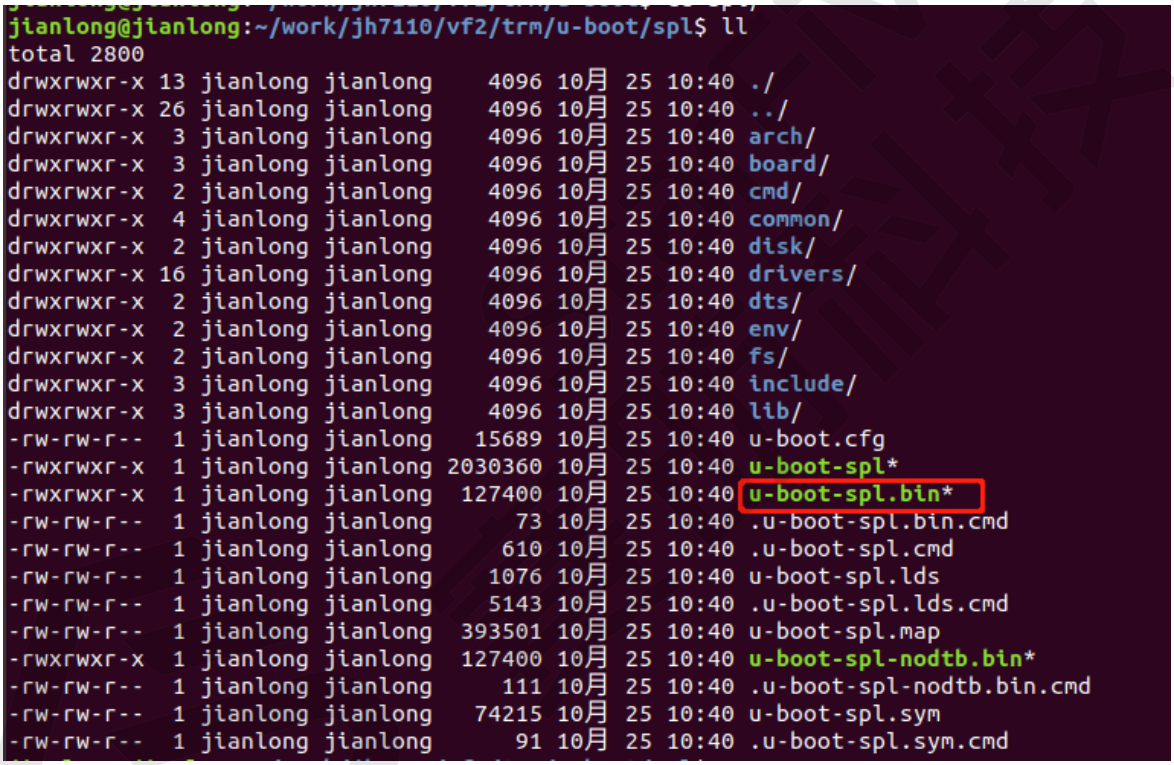


Figure 2-4 Example Output - u-boot-spl.bin



Tip:
Both `starfive_visionfive2.dtb` and `u-boot.bin` will be used later for OpenSBI compilation.

Tip:
`u-boot-spl.bin` will be used later for creating SPL file.

2.1.3. Creating SPL File

Follow the steps below to create the SPL file for VisionFive 2.

1. Locate to your desired directory to store the tools files. For example, the home directory.

   Example:

   ```
   cd ~ # home directory
   ```

2. Download the source code for U-Boot compilation.

   ```
   git clone https://github.com/starfive-tech/Tools
   ```

3. Switch to the code branch by executing the following command:

   ```
   cd Tools
   git checkout master
   git pull
   ```

4. Type the following to generate SPL tool under the `spl_tool` directory.

   ```
   cd spl_tool/
   make
   ```

   Figure 2-5 Example Output

   ```
   yingpeng@ubuntu:~/workspace/JH7110/github/Tools/spl_tool$ make
   cc -Wall -Wno-unused-result -Wno-format-truncation -O2 -c -o crc32.o crc32.c
   cc -Wall -Wno-unused-result -Wno-format-truncation -O2 -c -o spl_tool.o spl_tool.c
   cc -Wall -Wno-unused-result -Wno-format-truncation -O2  crc32.o spl_tool.o -o spl_tool
   yingpeng@ubuntu:~/workspace/JH7110/github/Tools/spl_tool$ ls
   crc32.c  crc32.o  LICENSE  Makefile  README.md  spl_tool  spl_tool.c  spl_tool.o
   yingpeng@ubuntu:~/workspace/JH7110/github/Tools/spl_tool$
   ```

5. Type the following to generate SPL file:

   ```
   ./spl_tool -c -f ${U_BOOT_PATH}/spl/u-boot-spl.bin
   ```

   Tip:
   Modify the *{U_BOOT_PATH}* to the path of u-boot from before.

Result:

You will see a new file named `u-boot-spl.bin.normal.out` generated under `{U_BOOT_PATH}/spl`. Refer to *Updating SPL and U-Boot* section in *VisionFive 2 Single Board Computer Quick Start Guide* to flash `u-boot-spl.bin.normal.out`.

Figure 2-6 Example Output

```
yingpeng@ubuntu:~/workspace/JH7110/github/Tools/spl_tool$ ./spl_tool -c -f /home/yingpeng/workspace/JH7110/github/u-boot/spl/u-boot-spl.bin
ubsplhdr.sofs:0x240, ubsplhdr.bofs:0x200000, ubsplhdr.vers:0x1010101 name:/home/yingpeng/workspace/JH7110/github/u-boot/spl/u-boot-spl.bin
SPL written to /home/yingpeng/workspace/JH7110/github/u-boot/spl/u-boot-spl.bin.normal.out successfully.
yingpeng@ubuntu:~/workspace/JH7110/github/Tools/spl_tool$ ls /home/yingpeng/workspace/JH7110/github/u-boot/spl/ -ll
total 2912
drwxrwxr-x  3 yingpeng yingpeng    4096 Mar  1 10:55 arch
drwxrwxr-x  3 yingpeng yingpeng    4096 Mar  1 10:55 board
drwxrwxr-x  2 yingpeng yingpeng    4096 Mar  1 10:55 cmd
drwxrwxr-x  4 yingpeng yingpeng    4096 Mar  1 10:55 common
drwxrwxr-x  2 yingpeng yingpeng    4096 Mar  1 10:55 disk
drwxrwxr-x 16 yingpeng yingpeng    4096 Mar  1 10:55 drivers
drwxrwxr-x  2 yingpeng yingpeng    4096 Mar  1 10:55 dts
drwxrwxr-x  2 yingpeng yingpeng    4096 Mar  1 10:55 env
drwxrwxr-x  2 yingpeng yingpeng    4096 Mar  1 10:55 fs
drwxrwxr-x  3 yingpeng yingpeng    4096 Mar  1 10:55 include
drwxrwxr-x  3 yingpeng yingpeng    4096 Mar  1 10:55 lib
-rw-rw-r--  1 yingpeng yingpeng   16252 Mar  1 10:54 u-boot.cfg
-rwxrwxr-x  1 yingpeng yingpeng 2043128 Mar  1 10:55 u-boot-spl
-rwxrwxr-x  1 yingpeng yingpeng  130240 Mar  1 10:55 u-boot-spl.bin
-rw-rw-r--  1 yingpeng yingpeng  131264 Mar  1 14:54 u-boot-spl.bin.normal.out
-rw-rw-r--  1 yingpeng yingpeng    1076 Mar  1 10:55 u-boot-spl.lds
-rw-rw-r--  1 yingpeng yingpeng  395008 Mar  1 10:55 u-boot-spl.map
-rwxrwxr-x  1 yingpeng yingpeng  130240 Mar  1 10:55 u-boot-spl-nodtb.bin
-rw-rw-r--  1 yingpeng yingpeng   74875 Mar  1 10:55 u-boot-spl.sym
yingpeng@ubuntu:~/workspace/JH7110/github/Tools/spl_tool$
```

2.1.4. Compiling OpenSBI

OpenSBI stands for Open-source Supervisor Binary Interface and it is an open-source implementation of the RISC-V Supervisor Binary Interface. It is a RISC-V-specific runtime service provider and it is typically used in boot stage following ROM and LOADER. A typical boot flow is as follows:

Figure 2-7 Typical Boot Flow

Follow the steps below to compile OpenSBI for VisionFive 2.

1. Locate to your desired directory to store the OpenSBI files. For example, the home directory.

   ```
   cd ~ # home directory
   ```

2. Download the source code for OpenSBI compilation.

   ```
   git clone https://github.com/starfive-tech/opensbi.git
   ```

3. Inside `opensbi` directory, type the following to compile openSBI.

   ```
   cd opensbi
   make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
    PLATFORM=generic FW_PAYLOAD_PATH=${U_BOOT_PATH}/u-boot.bin
    FW_FDT_PATH=${U_BOOT_PATH}/arch/riscv/dts/starfive_visionfive2.dtb
    FW_TEXT_START=0x40000000
   ```

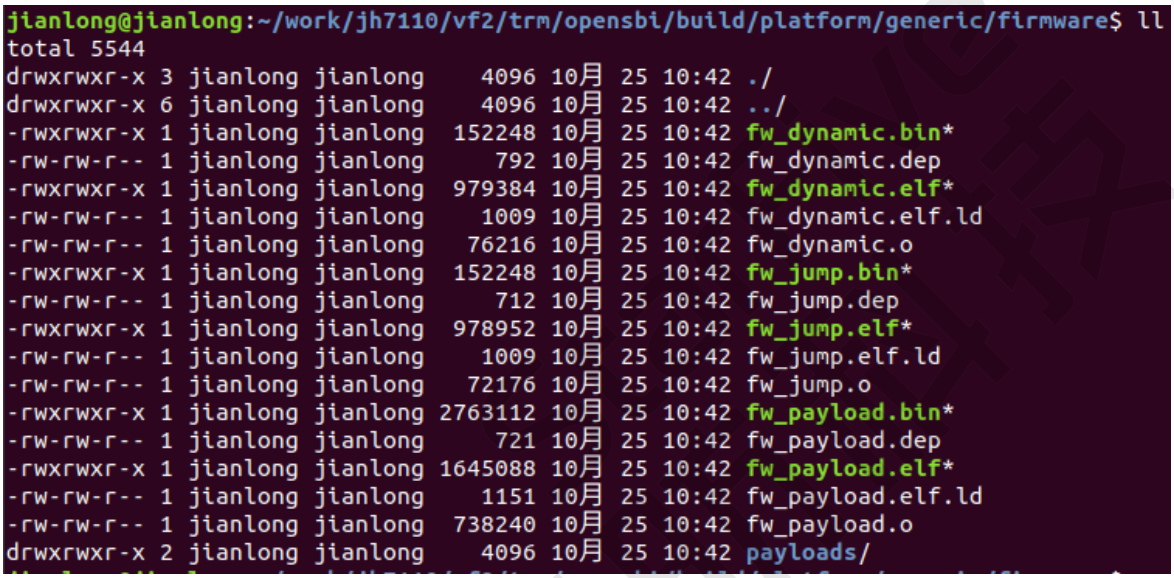
Tip:

Modify the *{U_BOOT_PATH}* to the path of U-Boot from before.

Result:

After compilation, the file `fw_payload.bin` will be generated in the directory `opensbi/build/platform/generic/firmware` and the size is larger than 2M.

Figure 2-8 Example Output



2.1.5. Creating fw_payload File

Follow the steps below to create the `fw_payload` for VisionFive 2.

1. Locate to the tools directory, which git clone from before.

```
cd Tools/uboot_its
```

2. Copy the output file `fw_payload.bin` from the OpenSBI compilation to the tools path:

```
cp ${OPENSBI_PATH}/build/platform/generic/firmware/fw_payload.bin ./
```

Note:

Modify the *{OPENSBI_PATH}* to the path of OpenSBI before executing.

3. Type the following to create `fw_payload` file under the `uboot_its` directory.

```
${U_BOOT_PATH}/tools/mkimage -f visionfive2-uboot-fit-image.its -A
 riscv -O u-boot -T firmware visionfive2_fw_payload.img
```

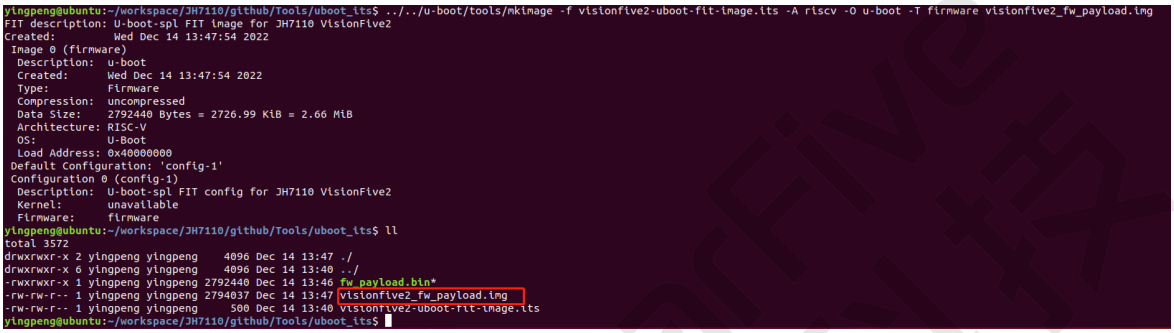
Note:

Remove the line break when copying this command from PDF.

Result:

You will see a new file named `visionfive2_fw_payload.img` generated. Refer to *Updating SPL and U-Boot* section in *VisionFive 2 Single Board Computer Quick Start Guide* to flash `visionfive2_fw_payload.img`.

Figure 2-9 Example Output



2.2. Compile and Update Linux Kernel

This chapter introduces the following sections:

2.2.1. Obtaining OS Version (Debian OS)

Steps:

1. Visit this link to download the latest operating system.
2. Flash the latest operating system to the Micro-SD card. For details, see *Flashing OS to a Micro-SD Card* section in *VisionFive 2 Single Board Computer Quick Start Guide*.

2.2.2. Software Environment

Follow the steps below to set up the software environment:

1. Execute the following command to compile the component:

```
$ sudo apt-get install build-essential linux-source bc kmod cpio
 flex libncurses5-dev libelf-dev libssl-dev dwarves bison git
 gcc-riscv64-linux-gnu g++-riscv64-linux-gnu vim tree
```

2. Execute the following command to checkou source code:

```
$ git clone https://github.com/starfive-tech/linux.git
```

3. See the Debian release information, find and switch the kernel source code to the corresponding version. In this section, taking Debian202403 as an example, the corresponding kernel version is v5.11.3. Execute the following command to switch branch:

```
$ git checkout JH7110_VF2_515_v5.11.3
```

The following figure is an example output:

Figure 2-10 Example Output

4. Execute the following command to set the default configuration of compiling Linux kernel:

```
make <Configuration_File> CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
```

Tip:
<Configuration_File>: This file is `starfive_visionfive2_defconfig` on VisionFive 2.

5. (Optional) Modify the configuration file. To modify the kernel configuration, execute the following command:

```
$ make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- menuconfig
```

The following is an example of modifying a configuration file:

Change the CPUFreq governor from **ondemand** to **performance**.

Under **Devices Drivers > CPU Frequency scaling** , change the Default CPUFreq governor from **ondemand** to **performance** and uncheck the **ondemand** option, as shown in the following figure:

Figure 2-11 Ondemand

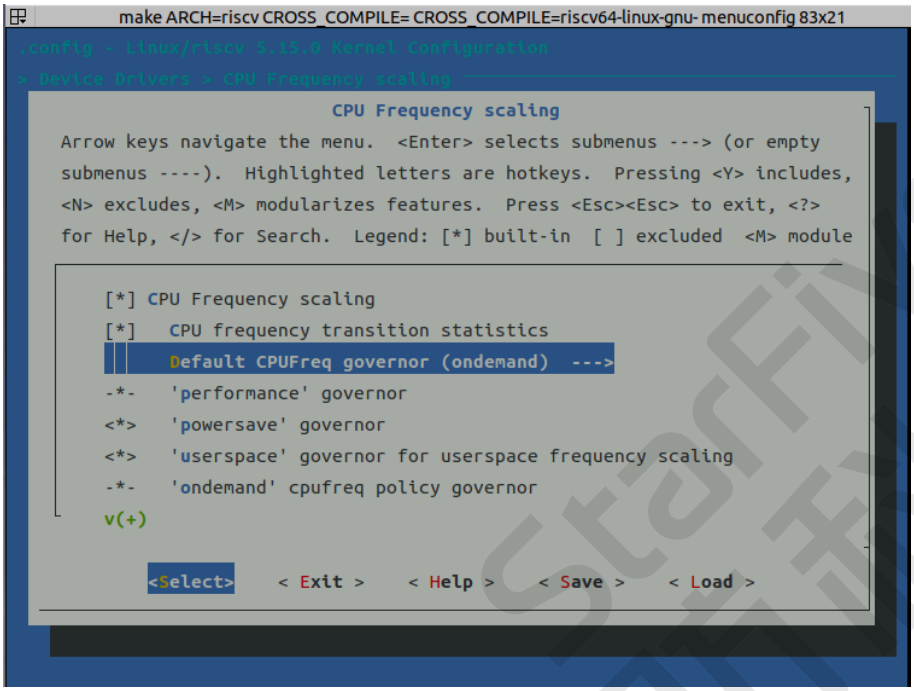
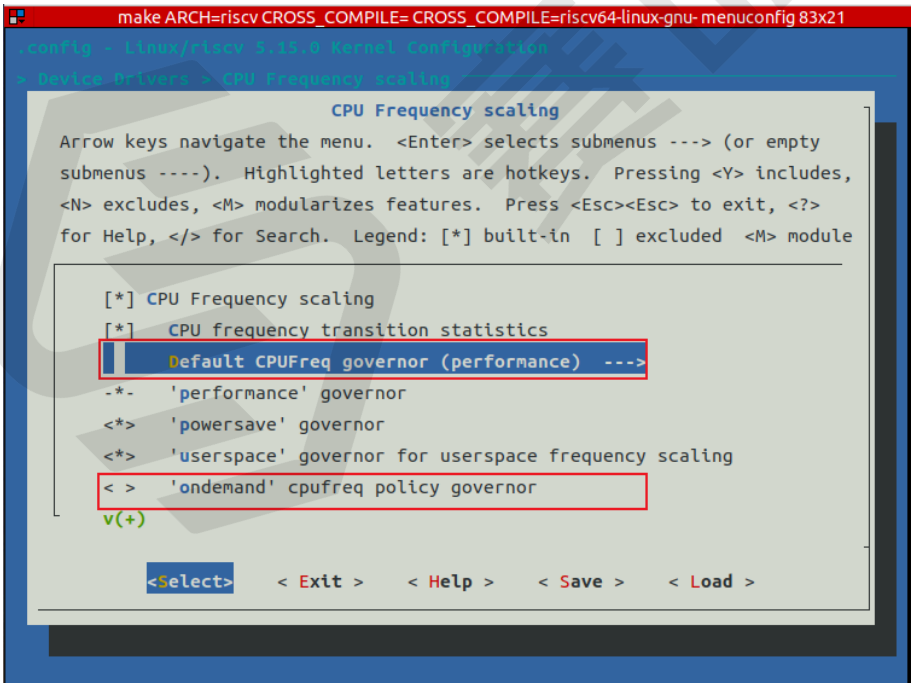


Figure 2-12 Performance



2.2.3. Compile Debian and Update Kernel

Execute the first 3 steps in Software Environment *(on page 16)*, and choose the compile method according to your compilation environment:

- Local compile and Install *(on page 19)*
- Cross compile and Install *(on page 19)*

Local compile and Install

1. Use `bindeb-pkg` to creat kernel:

```
cd linux/
cp arch/riscv/configs/starfive_visionfive2_defconfig .config
make ARCH=riscv olddefconfig
make ARCH=riscv -j$(nproc) bindeb-pkg
```

2. After compilation, install the `.deb` kernel package.

```
dpkg -i *.deb
```

Cross compile and Install

Tip:
Click the following link for reference:

- https://wiki.debian.org/BuildADebianKernelPackage
- https://www.debian.org/doc/manuals/debian-handbook/sect.kernel-compilation.zh-cn.html

1. Execute the following command to set the default configuration of compiling Linux kernel:

```
make <Configuration_File> CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
```

Tip:
<Configuration_File>: This file is `starfive_visionfive2_defconfig` on VisionFive 2.

2. Execute the following command to compile the kernel image and header files, and then package them as Debian packages:

```
$ nice make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- bindeb-pkg
 -j$(nproc)
KDEB_COMPRESS=xz LOCALVERSION='local_version'
```

Tip:

`local_version` is the version of the compiled kernel, which sets to `-performance` in this example, so the whole command is:

```
$ nice make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
 bindeb-pkg -j$(nproc)
KDEB_COMPRESS=xz LOCALVERSION=-performance
```

3. After compilation, the following files and `dtb` files will be generated in the upper directory:

Figure 2-13 File

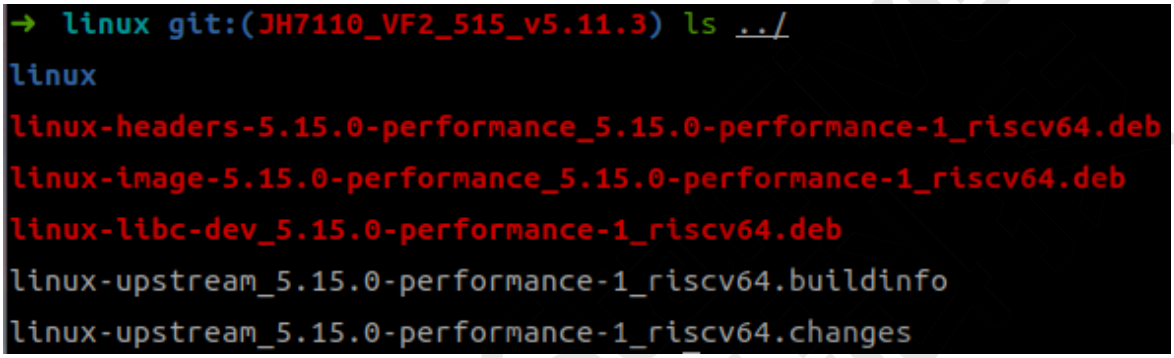
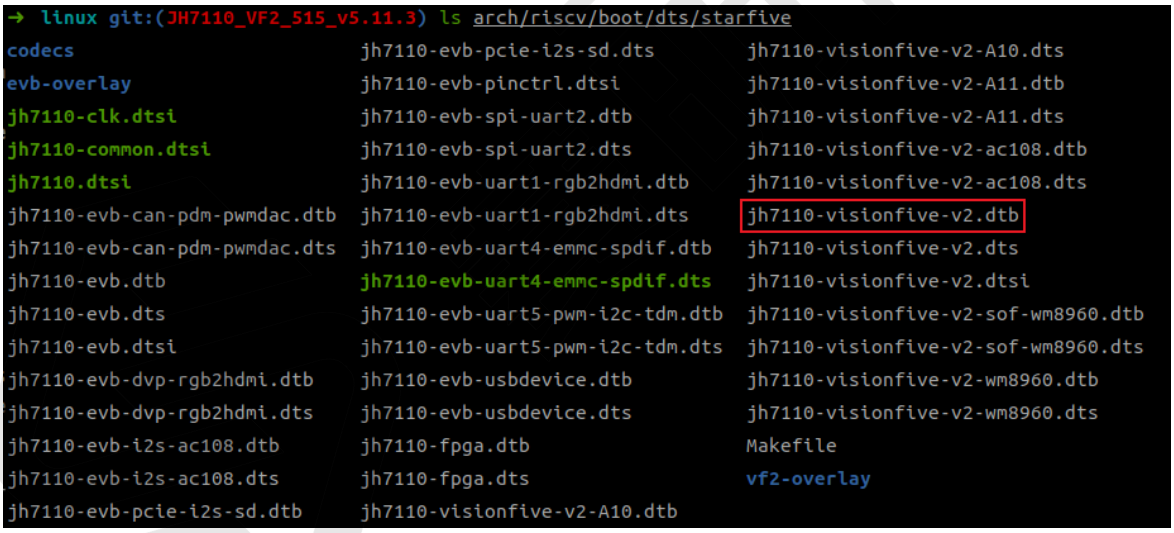
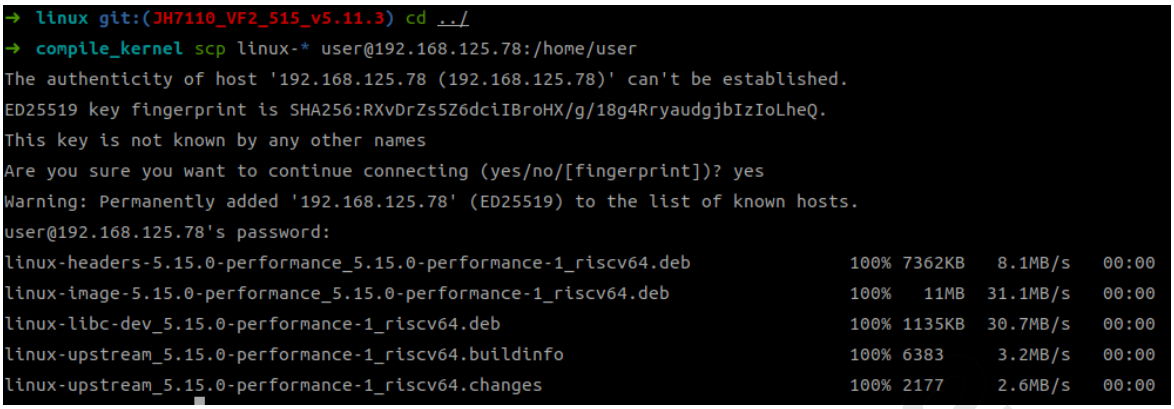


Figure 2-14 Dtb Files



4. Transfer the compiled Debian package and `dtb` files through network (SCP) or portable storage medium (USB) to VisionFive 2. The following figure shows an example output of file transfer over the network:

Figure 2-15 Example Output

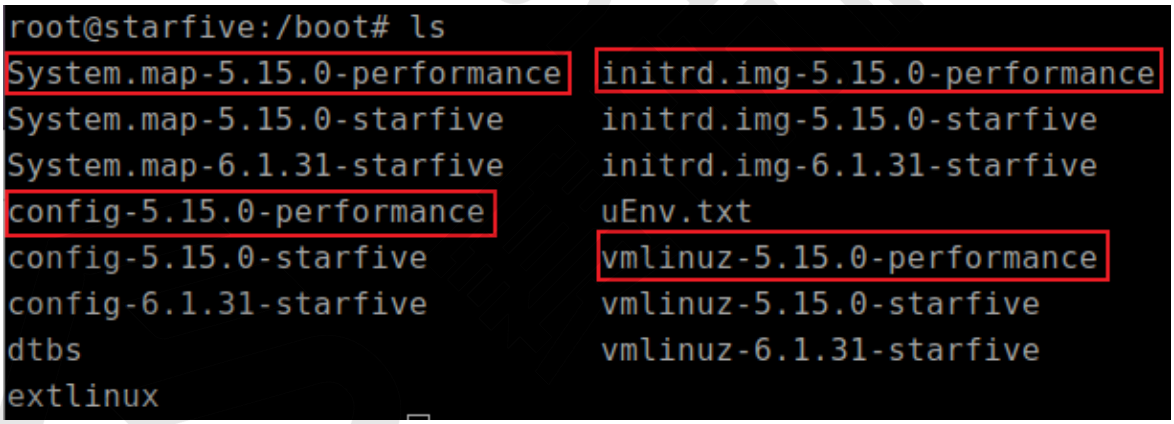


5. Execute the following command to install Debian:

```
$ dpkg -i
 linux-headers-5.15.0-performance_5.15.0-performance-1_riscv64.deb
$ dpkg -i
 linux-image-5.15.0-performance_5.15.0-performance-1_riscv64.deb
$ dpkg -i linux-libc-dev_5.15.0-performance-1_riscv64.deb
```

6. After installation, the files under `/boot` will be updated into:

Figure 2-16 Files under `/boot`



2.2.3.1. Update Configuration Files

VisionFive 2Previously, the internal boot mechanism of the Debian image has undergone several modifications (mainly about the loading address of `dtb` file), so it is necessary to explain the kernel configuration updates of different versions about Debian image:

- Debian202403 *(on page 21)*
- Debian202302 - Debian202311 *(on page 27)*

2.2.3.1.1. Debian202403

Update Configuration Files

Follow the steps below to update the configuration file of the Debian202403 version image:

1. Before installing the compiled files, the structures of `extlinux/extlinux.cnf`, `uEnv.txt`, and `dtbs/` under `/boot` path are as follows:

Figure 2-17 Directory Structure

```
root@starfive:/boot# cat extlinux/extlinux.conf
## /extlinux/extlinux.conf
##
## IMPORTANT WARNING
##
## The configuration of this file is generated automatically.
## Do not edit this file manually, use: u-boot-update

default l0
menu title U-Boot menu
prompt 0
timeout 50


label l0
        menu label Debian GNU/Linux bookworm/sid 6.1.31-starfive
        linux /vmlinuz-6.1.31-starfive
        initrd /initrd.img-6.1.31-starfive


        fdtdir /dtbs/6.1.31
        append  root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0

label l0r
        menu label Debian GNU/Linux bookworm/sid 6.1.31-starfive (rescue target)
        linux /vmlinuz-6.1.31-starfive
        initrd /initrd.img-6.1.31-starfive

        fdtdir /dtbs/6.1.31
        append  root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0
single


label l1
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive


        fdtdir /dtbs/5.15.0
        append  root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0

label l1r
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive (rescue target)
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive

        fdtdir /dtbs/5.15.0
        append  root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0
single
```

Figure 2-18 Directory Structure

```
root@starfive:/boot# cat uEnv.txt
fdt_high=0xffffffffffffffff
initrd_high=0xffffffffffffffff
kernel_addr_r=0x40200000
kernel_comp_addr_r=0x5a000000
kernel_comp_size=0x4000000
fdt_addr_r=0x46000000
ramdisk_addr_r=0x46100000
# Move distro to first boot to speed up booting
boot_targets=distro mmc0 dhcp
# Fix wrong fdtfile name
fdtfile=starfive/jh7110-visionfive-v2.dtb
# Fix missing bootcmd
bootcmd=run load_distro_uenv;run bootcmd_distro
```

Figure 2-19 Directory Structure

```
root@starfive:/boot# tree ./dtbs -L 2
./dtbs
├── 5.15.0
│   ├── sifive
│   └── starfive
└── 6.1.31
    ├── sifive
    └── starfive

6 directories, 0 files
```

2. After installation, add the following files to `/boot` path:

Figure 2-20 Added Files

```
root@starfive:/boot# ls
System.map-5.15.0-performance  extlinux
System.map-5.15.0-starfive     initrd.img-5.15.0-performance
System.map-6.1.31-starfive     initrd.img-5.15.0-starfive
config-5.15.0-performance      initrd.img-6.1.31-starfive
config-5.15.0-starfive         uEnv.txt
config-6.1.31-starfive         vmlinuz-5.15.0-performance
dtbs                           vmlinuz-5.15.0-starfive
dtbs-performance               vmlinuz-6.1.31-starfive
```

At the same time, the `extLinux/extlinux.conf` file has been modified, as shown in the following figure:

Figure 2-21 extlinux/extlinux.conf

```
root@starfive:/boot# cat extlinux/extlinux.conf
## /boot/extlinux/extlinux.conf
##
## IMPORTANT WARNING
##
## The configuration of this file is generated automatically.
## Do not edit this file manually, use: u-boot-update

default l0
menu title U-Boot menu
prompt 0
timeout 50


label l0
        menu label Debian GNU/Linux bookworm/sid 6.1.31-starfive
        linux /vmlinuz-6.1.31-starfive
        initrd /initrd.img-6.1.31-starfive


        fdtdir /dtbs
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0

label l0r
        menu label Debian GNU/Linux bookworm/sid 6.1.31-starfive (rescue target)
        linux /vmlinuz-6.1.31-starfive
        initrd /initrd.img-6.1.31-starfive

        fdtdir /dtbs
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0 single


label l1
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive


        fdtdir /dtbs
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0

label l1r
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive (rescue target)
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive

        fdtdir /dtbs
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0 single
```

Figure 2-22 extlinux/extlinux.conf

```
label l2
        menu label Debian GNU/Linux bookworm/sid 5.15.0-performance
        linux /vmlinuz-5.15.0-performance
        initrd /initrd.img-5.15.0-performance


        fdtdir /dtbs
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0

label l2r
        menu label Debian GNU/Linux bookworm/sid 5.15.0-performance (rescue target)
        linux /vmlinuz-5.15.0-performance
        initrd /initrd.img-5.15.0-performance

        fdtdir /dtbs
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0 single
```

3. It can be seen that the `fdtdir` in each startup item is set to `/dtbs`. Combined with the configuration in `uEnv.txt`, it can be seen that after powering on, the device tree file will be loaded according to the path of `/boot/dtbs/starfive/jh7110-visionfive-v2.dtb`, which obviously does not match the directory structure under the `dtbs/` path. Therefore, the `fdtdir` for the startup options of `label l0`, `label l0r`, `label l1`, and `label l1r` should be modified to the state before installing the Debian package:

Figure 2-23 Modify fdtdir

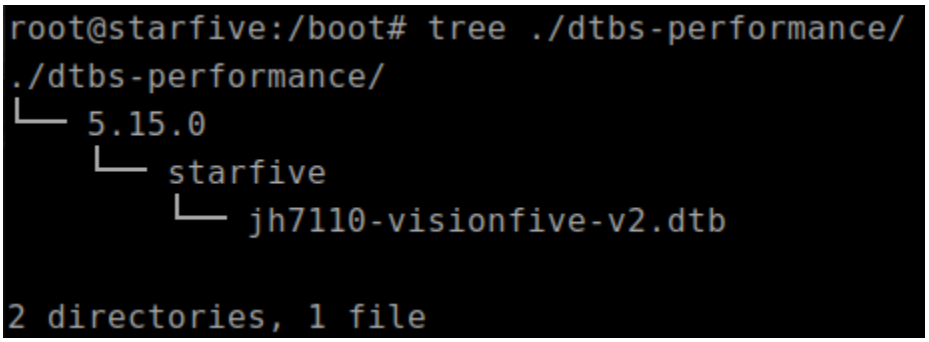
```
root@starfive:/boot# cat extlinux/extlinux.conf
## /boot/extlinux/extlinux.conf
##
## IMPORTANT WARNING
##
## The configuration of this file is generated automatically.
## Do not edit this file manually, use: u-boot-update

default l0
menu title U-Boot menu
prompt 0
timeout 50


label l0
        menu label Debian GNU/Linux bookworm/sid 6.1.31-starfive
        linux /vmlinuz-6.1.31-starfive
        initrd /initrd.img-6.1.31-starfive


        fdtdir /dtbs/6.1.31
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0

label l0r
        menu label Debian GNU/Linux bookworm/sid 6.1.31-starfive (rescue target)
        linux /vmlinuz-6.1.31-starfive
        initrd /initrd.img-6.1.31-starfive

        fdtdir /dtbs/6.1.31
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0 single


label l1
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive


        fdtdir /dtbs/5.15.0
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0

label l1r
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive (rescue target)
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive

        fdtdir /dtbs/5.15.0
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0 single
```

4. Execute the following command to set a new `dtb` addressing path for the new kernel boot option:

```
$ mkdir -p /boot/dtbs-performance/5.15.0/starfive
```

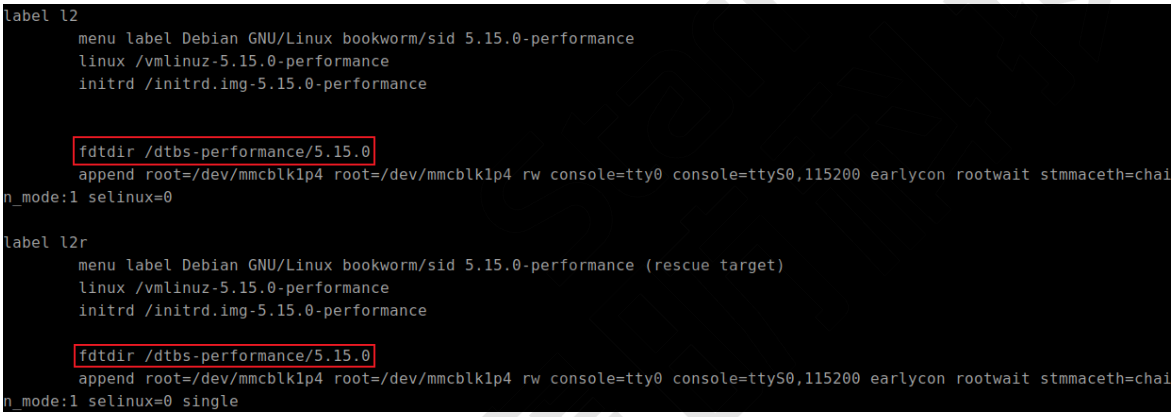
And place the device tree file under the compiled kernel source code in this path:

Figure 2-24 Place kernel source code



5. Modify the `fdtdir` configuration in `label l2` and `label l2r` in `extlinux/extlinux.conf` so that when starting a new kernel, the device tree file will be loaded from the `/boot/dtbs-performance/5.15.0/starfive/jh7110-visionfive-v2.dtb` path:

Figure 2-25 Modify fdtdir Configuration



Verfication

After replacing the kernel of Debian202403 image and powering it on, select the newly added kernel option in the U-Boot menu. As shown in the following two images, select the kernel boot options for 1 and 5 respectively to see that the corresponding device tree files have been correctly loaded:

Figure 2-26 Kernel Boot Option-1

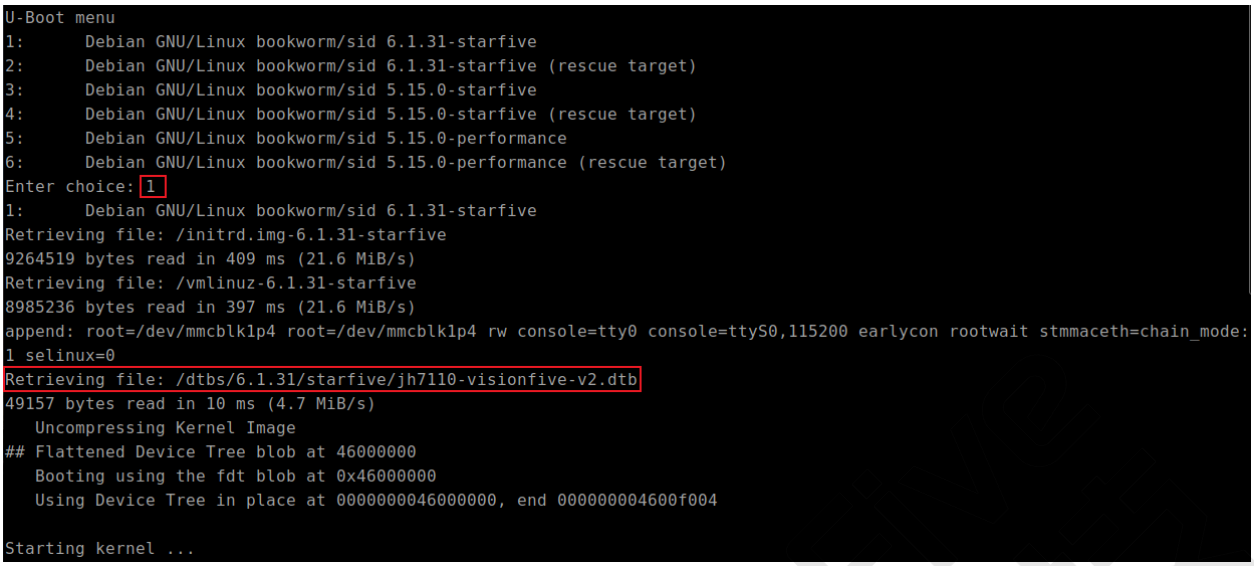
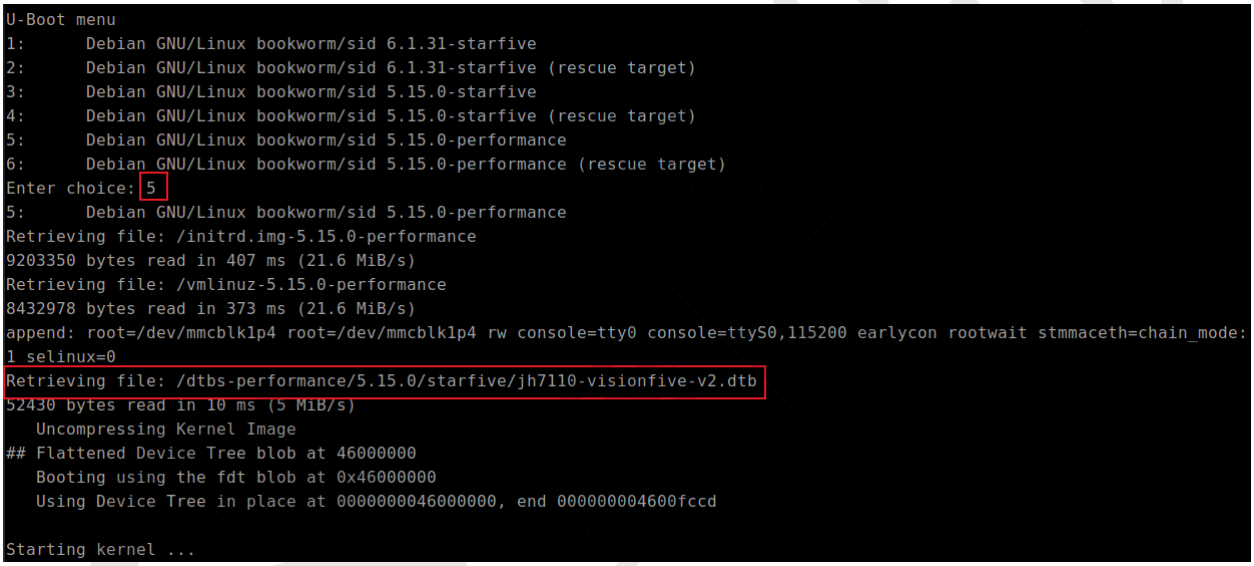


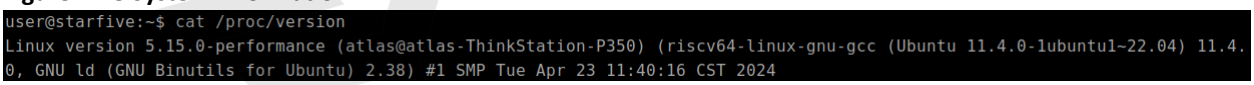
Figure 2-27 Kernel Boot Option-5



After logging in with power on, enter the following command to view system information:

```
$ cat /proc/version
```

Figure 2-28 System Information



2.2.3.1.2. Debian202302 - Debian202311

The boot configuration in Debian202302 image to Debian202311 image is the same, and replacing the kernel with Debian packages is also relatively simple. This section takes Debian202311 as an example:

Update Configuration Files

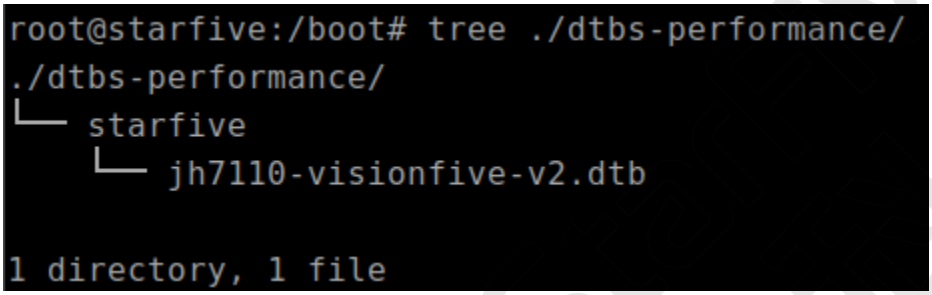
Follow the steps below to update the configuration file of the Debian202311 version image:

1. After compiling and installing the kernel Debian package, create the path to place the `dtb` files:
2. Execute the following command to set a new `dtb` addressing path for the new kernel boot option:

```
$ mkdir -p /boot/dtbs-performance/starfive
```

And place the device tree file under the compiled kernel source code in this path:

Figure 2-29 Place kernel source code



3. Modify `extlinux/extlinux.conf` file:

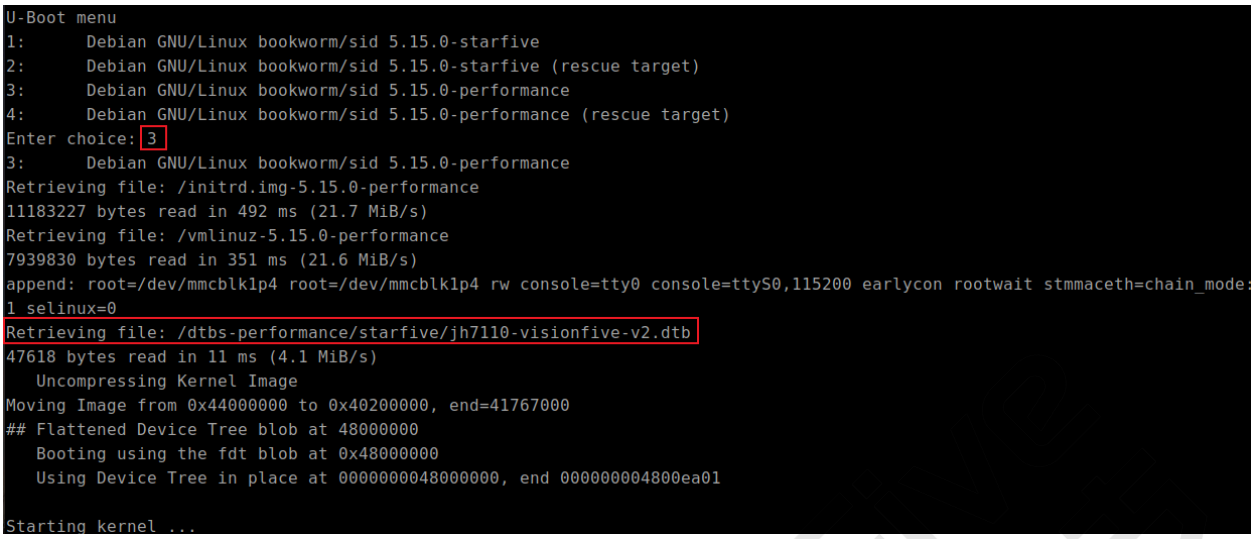
Figure 2-30 Modify File

```
root@starfive:/boot# cat extlinux/extlinux.conf
## /boot/extlinux/extlinux.conf
##
## IMPORTANT WARNING
##
## The configuration of this file is generated automatically.
## Do not edit this file manually, use: u-boot-update

default l0
menu title U-Boot menu
prompt 0
timeout 50


label l0
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive


        fdtdir /dtbs
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0

label l0r
        menu label Debian GNU/Linux bookworm/sid 5.15.0-starfive (rescue target)
        linux /vmlinuz-5.15.0-starfive
        initrd /initrd.img-5.15.0-starfive

        fdtdir /dtbs
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0 single


label l1
        menu label Debian GNU/Linux bookworm/sid 5.15.0-performance
        linux /vmlinuz-5.15.0-performance
        initrd /initrd.img-5.15.0-performance


        fdtdir /dtbs-performance
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0

label l1r
        menu label Debian GNU/Linux bookworm/sid 5.15.0-performance (rescue target)
        linux /vmlinuz-5.15.0-performance
        initrd /initrd.img-5.15.0-performance

        fdtdir /dtbs-performance
        append root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chai
n_mode:1 selinux=0 single
```

Verfication

After powering on, select the newly added kernel option in the U-Boot menu. As shown in the following two images, select the kernel boot options 3 to see that the corresponding device tree files have been correctly loaded:

Figure 2-31 Kernel Boot Option-3



2.2.4. Compile Kernel and Manually Replace Updated Files

This section introduces the following parts:

- Compile Kernel, Device Tree, and Driver Module *(on page 30)*
- Replace Kernel and Update Configuration Files *(on page 32)*

2.2.4.1. Compile Kernel, Device Tree, and Driver Module

Follow the steps below to compile kernel, device tree, and driver module:

1. Execute the following command to set the default configuration of compiling Linux kernel:

   ```
   make <Configuration_File> CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
   ```

   Tip:
   <Configuration_File>: This file is `starfive_visionfive2_defconfig` on VisionFive 2.

2. After setting up the software environment, execute the following command to compile the source code:

   ```
   $ make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv -j$(nproc)
   ```

3. Execute the following command to create a directory for storing the generated kernel files:

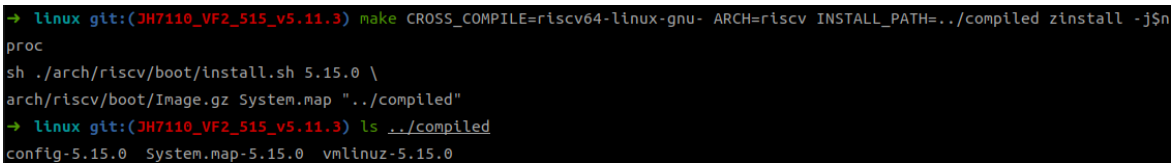
   ```
   $ mkdir ../compiled
   ```

4. Execute the following command to compile and generate files such as `config`, `system.map`, and `vmlinuz` to the specified path:

```
$ make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
 INSTALL_PATH=../compiled zinstall -j$(nproc)
```

The following figure is an example output:

Figure 2-32 Example Output



5. Execute the following command to copy `dtb` files:

```
$ cp arch/riscv/boot/dts/starfive/jh7110-visionfive-v2.dtb ../compiled
```

6. (Optional) Execute the following command to compile and generate files to the specified path:

```
$ make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
 INSTALL_MOD_PATH=../compiled modules_install
```

Note:
If the new kernel does not involve changes to the driver module, it can be omitted and the default kernel's driver module will still be used.

The following figure is an example output:

Figure 2-33 Example Output

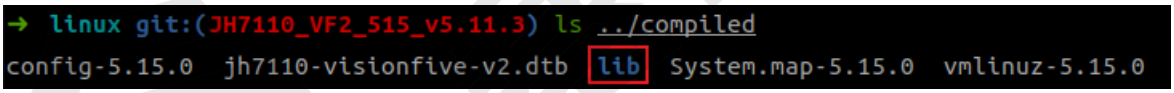
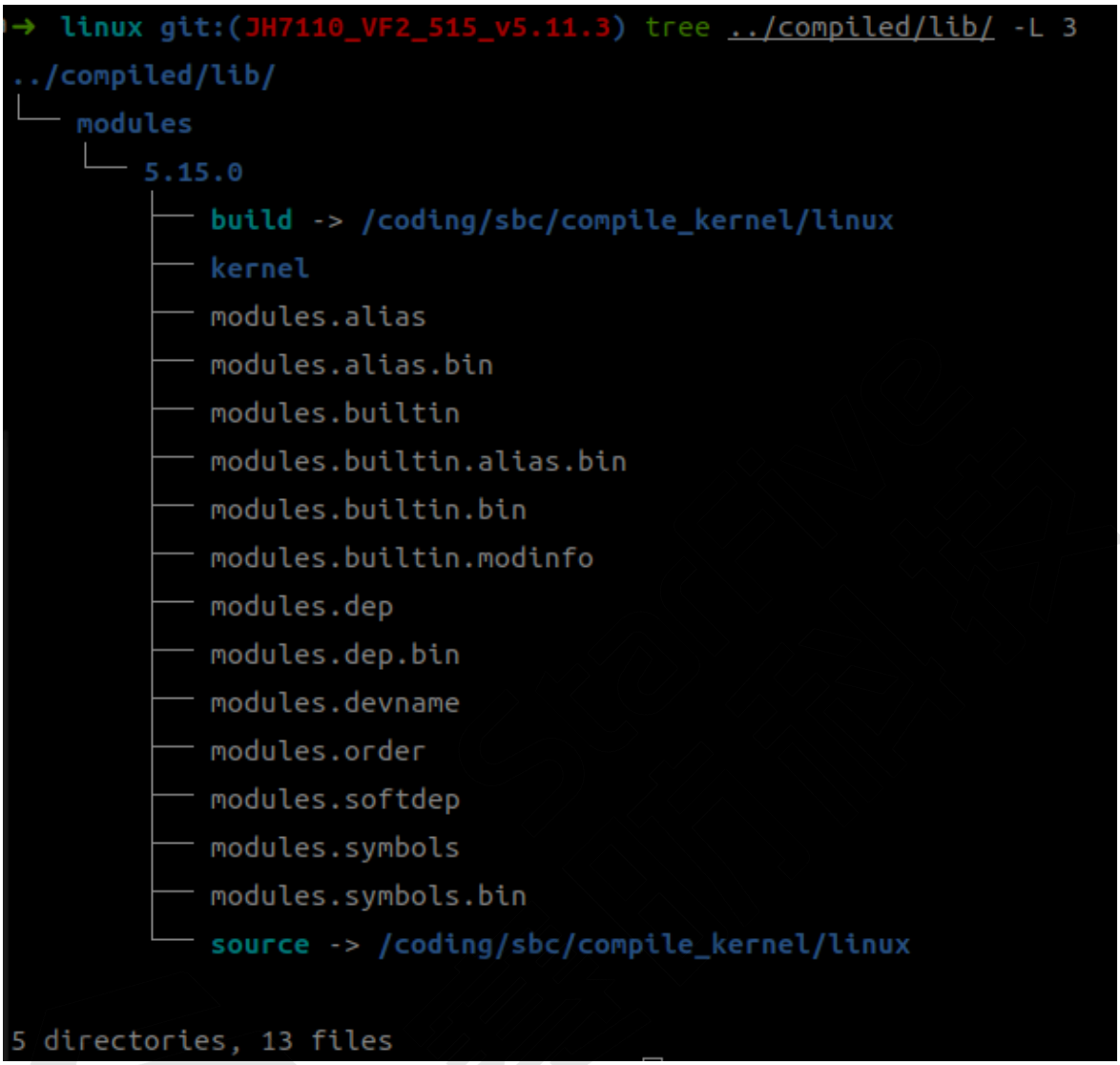


Figure 2-34 Example Output



Note:
There is no link between `build` and `source` in the corresponding path of the Debian image. The corresponding item in the above figure can be deleted.

2.2.4.2. Replace Kernel and Update Configuration Files

This section introduces the following parts:

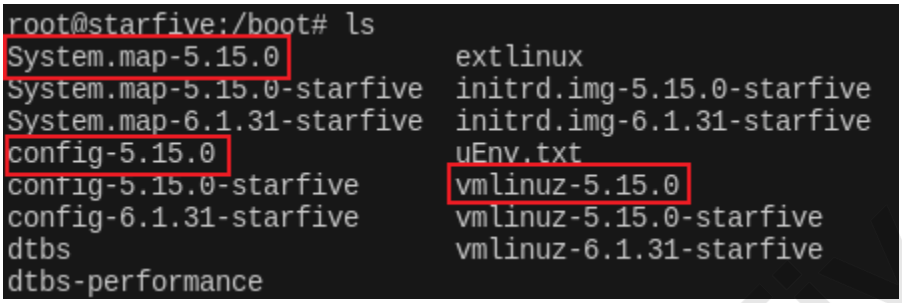
- Replace Kernel *(on page 32)*
- Update Configuration Files *(on page 34)*

2.2.4.2.1. Replace Kernel

Place the files generated by compiling under `compiled/` path onto the Debian running system via a network or removable storage medium, and place each file in the corresponding path.

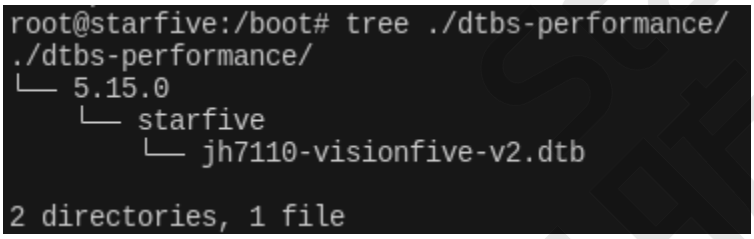
1. Place the files `System.map-5.15.0`, `config-5.15.0` and `vmlinuz-5.15.0` under `/boot` path:

Figure 2-35 Place Files



2. The device tree file can refer to the default path, creates a new path corresponding to the version (`/boot/dtbs performance/5.15.0/starfive` in this example), and place `jh7110 visionfive v2.dtb` in it:

Figure 2-36 Place Files



3. (Optional) Place the files under `/lib/modules` to `/lib/modules` of VisionFive 2:

Figure 2-37 Place Files

4. (Optional) Enter the corresponding version of the kernel module path and execute the following command to generate `initramfs`:

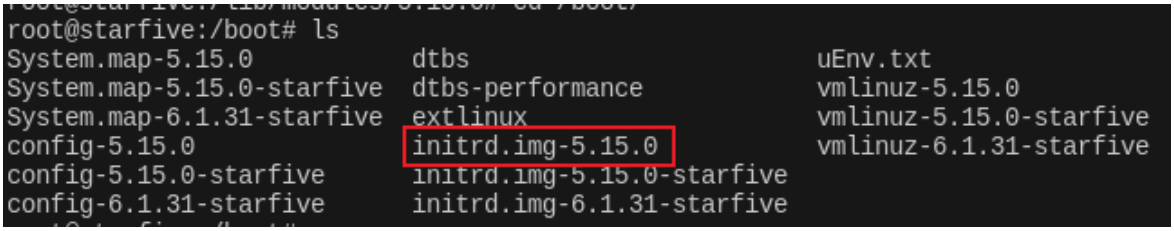
```
update-initrsmfs -c -k 5.15.0 -b /boot
```

Figure 2-38 Generate initramfs

Result:

Generate an `initrd.img` file with the corresponding version number under `/boot` path.

Figure 2-39 initrd.img



Note:

`update-initramfs` is a command used to generate `initramfs` (initial memory file system). `-xxx` parameter specifies which kernel version to generate `initramfs` for. You need to replace `-xxx` with the actual kernel version number of the `initramfs` you want to generate (in this example is `5.15.0`).

2.2.4.2.2. Update Configuration Files

After replacing the above files, it is necessary to modify the `extlinux/extlinux.cn` file to add the boot entry for the new kernel:

Figure 2-40 Modify `extlinux/extlinux.conf` file

```
label l2
        menu label Debian GNU/Linux bookworm/sid 5.15.0-performance
        linux /vmlinuz-5.15.0
        initrd /initrd.img-5.15.0


        fdtdir /dtbs-performance/5.15.0
        append  root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycoo
n rootwait stmmaceth=chain_mode:1 selinux=0
```

Verification

Follow the steps below to verify:

1. After powering on again, the newly added startup options can be viewed in the U-Boot menu. After selecting the corresponding options, it can be seen that files such as initrd.img, vmLinux, and dtb are correctly loaded from the set path.

Figure 2-41 U-Boot Menu

```
U-Boot menu
1:      Debian GNU/Linux bookworm/sid 6.1.31-starfive
2:      Debian GNU/Linux bookworm/sid 6.1.31-starfive (rescue target)
3:      Debian GNU/Linux bookworm/sid 5.15.0-starfive
4:      Debian GNU/Linux bookworm/sid 5.15.0-starfive (rescue target)
5:      Debian GNU/Linux bookworm/sid 5.15.0-performance
Enter choice: 5
5:      Debian GNU/Linux bookworm/sid 5.15.0-performance
Retrieving file: /initrd.img-5.15.0
10167560 bytes read in 446 ms (21.7 MiB/s)
Retrieving file: /vmlinuz-5.15.0
8432841 bytes read in 370 ms (21.7 MiB/s)
append: root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycoo
Retrieving file: /dtbs-performance/5.15.0/starfive/jh7110-visionfive-v2.dtb
52430 bytes read in 11 ms (4.5 MiB/s)
   Uncompressing Kernel Image
## Flattened Device Tree blob at 46000000
   Booting using the fdt blob at 0x46000000
   Using Device Tree in place at 0000000046000000, end 000000004600fccd

Starting kernel ...
```

Figure 2-42 Version

```
root@starfive:~# cat /proc/version
Linux version 5.15.0 (atlas@atlas-ThinkStation-P350) (riscv64-linux-gnu-gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, GNU ld (G
NU Binutils for Ubuntu) 2.38) #2 SMP Wed Apr 24 14:35:12 CST 2024
```

2. In addition, as mentioned earlier, if the new kernel does not involve changes to the driver module, the modules_install command can be omitted and a replacement version of the initrd.img file can be generated, which can also start the kernel normally.

   Then modify the extlinuxconf file to add a startup entry and change the initrd configuration from the generated initrd.img-5.15.0 to the default initrd.img-5.15.0-starfive:

Figure 2-43 initrd.img-5.15.0-starfive

```
label l2
        menu label Debian GNU/Linux bookworm/sid 5.15.0-performance
        linux /vmlinuz-5.15.0
        initrd /initrd.img-5.15.0-starfive


        fdtdir /dtbs-performance/5.15.0
        append  root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycoo
n rootwait stmmaceth=chain_mode:1 selinux=0
```

Tip:
For `fdtdir` configuration, please refer to Update Configuration Files *(on page 21)*.

3. After powering on again and selecting the corresponding option in the `U-Boot menu`, it can be seen that `initrd.img-5.15.0-starfive` is loaded and the system starts normally:

Figure 2-44 U-Boot Menu

```
U-Boot menu
1:      Debian GNU/Linux bookworm/sid 6.1.31-starfive
2:      Debian GNU/Linux bookworm/sid 6.1.31-starfive (rescue target)
3:      Debian GNU/Linux bookworm/sid 5.15.0-starfive
4:      Debian GNU/Linux bookworm/sid 5.15.0-starfive (rescue target)
5:      Debian GNU/Linux bookworm/sid 5.15.0-performance
Enter choice: 5
5:      Debian GNU/Linux bookworm/sid 5.15.0-performance
Retrieving file: /initrd.img-5.15.0-starfive
9252487 bytes read in 406 ms (21.7 MiB/s)
Retrieving file: /vmlinuz-5.15.0
8432841 bytes read in 371 ms (21.7 MiB/s)
append: root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycoo
Retrieving file: /dtbs-performance/5.15.0/starfive/jh7110-visionfive-v2.dtb
52430 bytes read in 11 ms (4.5 MiB/s)
   Uncompressing Kernel Image
## Flattened Device Tree blob at 46000000
   Booting using the fdt blob at 0x46000000
   Using Device Tree in place at 0000000046000000, end 000000004600fccd

Starting kernel ...
```

Figure 2-45 Version

```
root@starfive:~# cat /proc/version
Linux version 5.15.0 (atlas@atlas-ThinkStation-P350) (riscv64-linux-gnu-gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, GNU ld (G
NU Binutils for Ubuntu) 2.38) #2 SMP Wed Apr 24 14:35:12 CST 2024
```

2.3. Replace the dtb Files to be Loaded

By enabling different device tree files, it is possible to achieve different functions or support different peripherals on VisionFive 2. Taking default kernel of Debian202403 image as an example, it supports the following different device tree files:

Figure 2-46 dtb Files

Note:

Different boards use different dtb files:

- `jh7110-visionfive-v2.dtb`: for Version 1.2A and 1.3B board.
- `jh7110-visionfive-v2-ac108.dtb`: for version 1.2A and 1.3B board with ac108 codec.
- `jh7110-visionfive-v2-tdm.dtb`: for version 1.2A and 1.3B board with tdm sound card.
- `jh7110-visionfive-wm8960.dtb`: for Version 1.2A and 1.3B board with wm8960 codec.

Modify `uEnv.txt` File

By modifying the `uEnv.txt` file, different device tree files can be loaded when the board starts. For example, modify the `uEnv.txt` file to use `jh7110-visionfive-v2-wm8960.dtb` to support the wm8960 codec:

Figure 2-47 Modify uEnv.txt File

```
root@starfive:/boot# cat uEnv.txt
fdt_high=0xffffffffffffffff
initrd_high=0xffffffffffffffff
kernel_addr_r=0x40200000
kernel_comp_addr_r=0x5a000000
kernel_comp_size=0x4000000
fdt_addr_r=0x46000000
ramdisk_addr_r=0x46100000
# Move distro to first boot to speed up booting
boot_targets=distro mmc0 dhcp
# Fix wrong fdtfile name
# fdtfile=starfive/jh7110-visionfive-v2.dtb
fdtfile=starfive/jh7110-visionfive-v2-wm8960.dtb
# Fix missing bootcmd
bootcmd=run load_distro_uenv;run bootcmd_distro
```

Verification

After re-powering on and selecting the corresponding kernel, it can be found that `jh7110-visionfive-v2-wm8960.dtb` has been correctly loaded from the corresponding path:

Figure 2-48 Verification

```
U-Boot menu
1:      Debian GNU/Linux bookworm/sid 6.1.31-starfive
2:      Debian GNU/Linux bookworm/sid 6.1.31-starfive (rescue target)
3:      Debian GNU/Linux bookworm/sid 5.15.0-starfive
4:      Debian GNU/Linux bookworm/sid 5.15.0-starfive (rescue target)
5:      Debian GNU/Linux bookworm/sid 5.15.0-performance
Enter choice: 1
1:      Debian GNU/Linux bookworm/sid 6.1.31-starfive
Retrieving file: /initrd.img-6.1.31-starfive
9264519 bytes read in 406 ms (21.8 MiB/s)
Retrieving file: /vmlinuz-6.1.31-starfive
8985236 bytes read in 395 ms (21.7 MiB/s)
append: root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0
Retrieving file: /dtbs/6.1.31/starfive/jh7110-visionfive-v2-wm8960.dtb
49982 bytes read in 11 ms (4.3 MiB/s)
   Uncompressing Kernel Image
## Flattened Device Tree blob at 46000000
   Booting using the fdt blob at 0x46000000
   Using Device Tree in place at 0000000046000000, end 000000004600f33d

Starting kernel ...
```

3. Making BusyBox System

This section describes how to make BusyBox system.

It contains the following sections:

3.1. Compile Linux (Cross Compile)

Follow the steps below to cross compile Linux:

1. Execute the following to install dependencies and create kernel:

```
apt-get install build-essential linux-source bc kmod cpio flex
 libncurses5-dev libelf-dev libssl-dev dwarves bison git
```

2. Checkout the kernel files from StarFive GitHub:

```
git clone https://github.com/starfive-tech/linux
```

3. Execute the following command to switch to code branch:

```
cd linux
git checkout -b JH7110_VisionFive2_devel
 origin/JH7110_VisionFive2_devel
git pull
```

4. Execute the following command to set the default configuration of compiling Linux kernel:

```
make <Configuration_File> CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
```

Tip:
<Configuration_File>: This file is `starfive_visionfive2_defconfig` on VisionFive 2.

5. Execute the following command to set other software configuration of compiling Linux kernel:

```
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv menuconfig
```

6. Compile Linux Kernel:

```
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv -jx
```

Note:

Here you need to change the *-jx* value according to the number of cores in your CPU. If your CPU has 8 cores, change this to *-j8*. This process will take some time and therefore please wait patiently. This process is quite time-consuming, please be patient and wait.

Result:

The system will generate the kernel image file `Image.gz` under `linux/arch/riscv/boot` directory.

Figure 3-1 Example Output

```
jianlong@jianlong:~/work/jh7110/vf2/trm/linux/arch/riscv/boot$ ll
total 21964
drwxrwxr-x  3 jianlong jianlong     4096 10月 26 11:03 ./
drwxrwxr-x 10 jianlong jianlong     4096 10月 26 11:01 ../
drwxrwxr-x  6 jianlong jianlong     4096 10月 26 11:00 dts/
-rw-rw-r--  1 jianlong jianlong       83 10月 26 11:00 .gitignore
-rwxrwxr-x  1 jianlong jianlong 22016512 10月 26 11:03 Image*
-rw-rw-r--  1 jianlong jianlong      151 10月 26 11:03 .Image.cmd
-rw-rw-r--  1 jianlong jianlong  7744843 10月 26 11:03 Image.gz
-rw-rw-r--  1 jianlong jianlong      101 10月 26 11:03 .Image.gz.cmd
-rw-rw-r--  1 jianlong jianlong     1561 10月 26 11:00 install.sh
-rw-rw-r--  1 jianlong jianlong      206 10月 26 11:00 loader.lds.S
-rw-rw-r--  1 jianlong jianlong      143 10月 26 11:00 loader.S
-rw-rw-r--  1 jianlong jianlong     1612 10月 26 11:00 Makefile
jianlong@jianlong:~/work/jh7110/vf2/trm/linux/arch/riscv/boot$
```

The system will generate the dtb file `Image.gz` under `linux/arch/riscv/boot` directory.

Figure 3-2 Generating dtb

```
jianlong@jianlong:~/work/jh7110/vf2/trm/linux/arch/riscv/boot/dts/starfive$ ll *.dtb
-rw-rw-r-- 1 jianlong jianlong 64849 10月 26 11:01 jh7110-evb-can-pdm-pwmdac.dtb
-rw-rw-r-- 1 jianlong jianlong 64498 10月 26 11:01 jh7110-evb.dtb
-rw-rw-r-- 1 jianlong jianlong 64249 10月 26 11:01 jh7110-evb-dvp-rgb2hdmi.dtb
-rw-rw-r-- 1 jianlong jianlong 64713 10月 26 11:01 jh7110-evb-i2s-ac108.dtb
-rw-rw-r-- 1 jianlong jianlong 65144 10月 26 11:01 jh7110-evb-pcie-i2s-sd.dtb
-rw-rw-r-- 1 jianlong jianlong 64369 10月 26 11:01 jh7110-evb-spi-uart2.dtb
-rw-rw-r-- 1 jianlong jianlong 64405 10月 26 11:01 jh7110-evb-uart1-rgb2hdmi.dtb
-rw-rw-r-- 1 jianlong jianlong 64907 10月 26 11:01 jh7110-evb-uart4-emmc-spdif.dtb
-rw-rw-r-- 1 jianlong jianlong 65005 10月 26 11:01 jh7110-evb-uart5-pwm-i2c-tdm.dtb
-rw-rw-r-- 1 jianlong jianlong 64353 10月 26 11:01 jh7110-evb-usbdevice.dtb
-rw-rw-r-- 1 jianlong jianlong 63510 10月 26 11:01 jh7110-fpga.dtb
-rw-rw-r-- 1 jianlong jianlong 47299 10月 26 11:01 jh7110-visionfive-v2-A10.dtb
-rw-rw-r-- 1 jianlong jianlong 47491 10月 26 11:01 jh7110-visionfive-v2-A11.dtb
-rw-rw-r-- 1 jianlong jianlong 48381 10月 26 11:01 jh7110-visionfive-v2-ac108.dtb
-rw-rw-r-- 1 jianlong jianlong 47743 10月 26 11:01 jh7110-visionfive-v2.dtb
-rw-rw-r-- 1 jianlong jianlong 48252 10月 26 11:01 jh7110-visionfive-v2-wm8960.dtb
```

When porting `rootfs`, `dtb`, and kernel to VisionFive 2, `Image.gz` and `.dtb` files will be used.

3.2. Making File System

Follow the following steps to make the file system.

1. Create the directory structure.

```
mkdir rootfs
cd rootfs
mkdir dev usr bin sbin lib etc proc tmp sys var root mnt
```

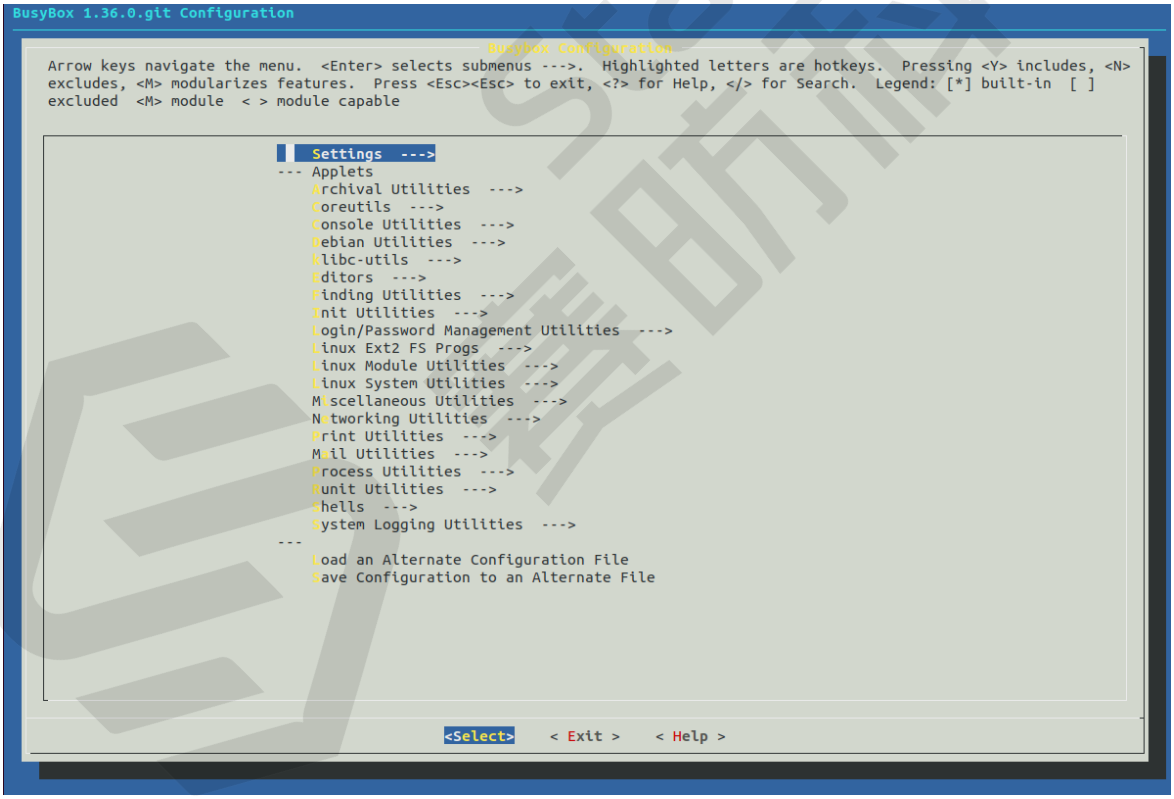
2. Download the BusyBox source code outside the rootfs directory.

```
git clone https://git.busybox.net/busybox
```

3. Navigate to the extracted location and enter BusyBox configuration.

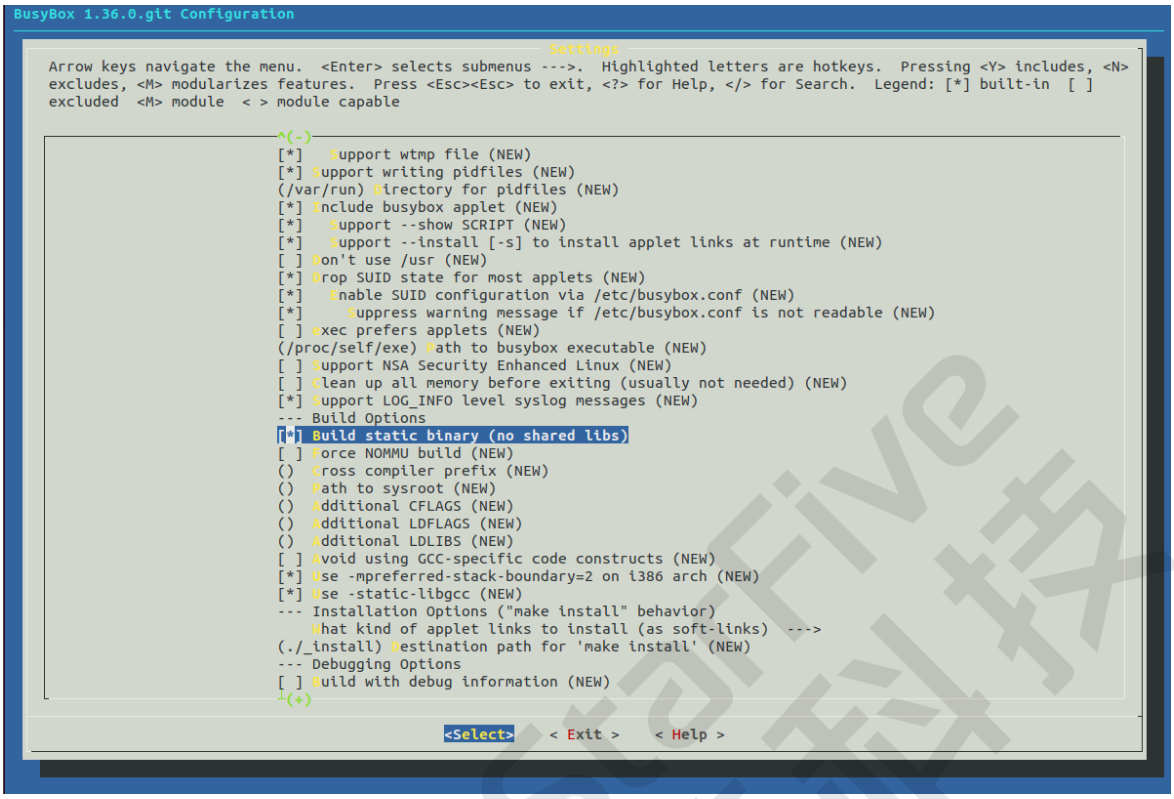
```
cd busybox
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv menuconfig
```

Figure 3-3 Busybox Configuration



4. Navigate to **Settings > Build Options** and check **Build static binary (no shared libs)** by pressing **Y.**

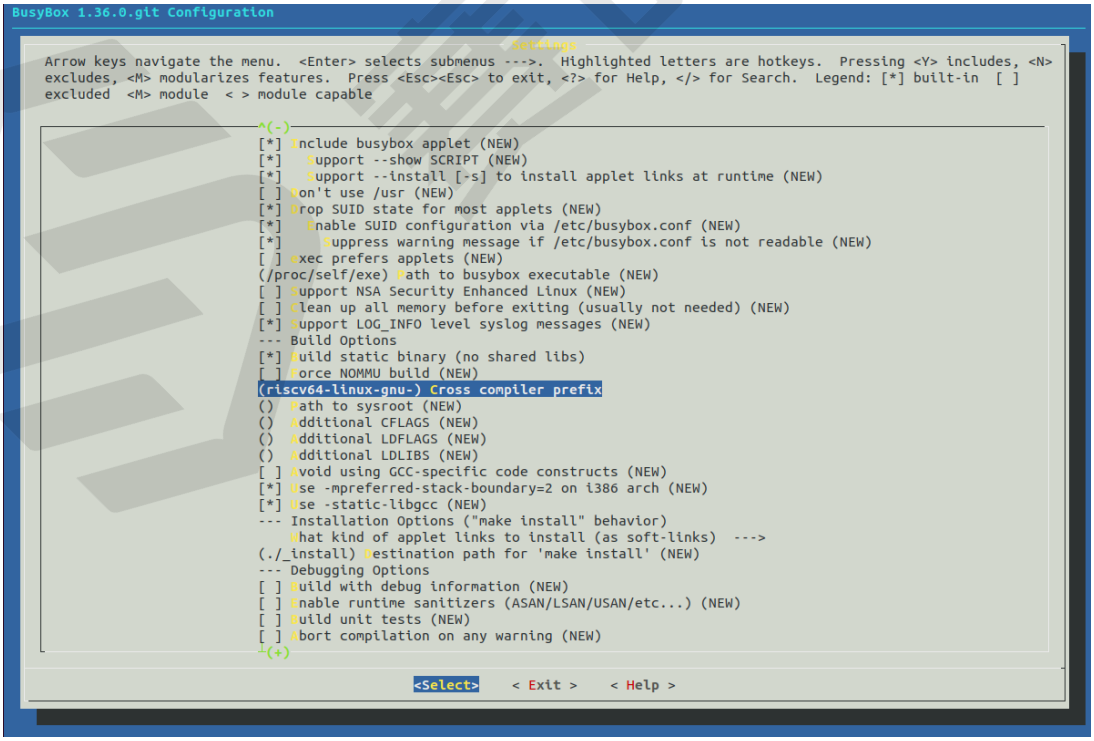
Figure 3-4 Check Build static binary (no shared libs)



5. Specify the compiler:

   a. Under **Build Options**, select **(riscv64-linux-gnu-) Cross compiler prefix**.

Figure 3-5 Select Cross Compiler Prefix



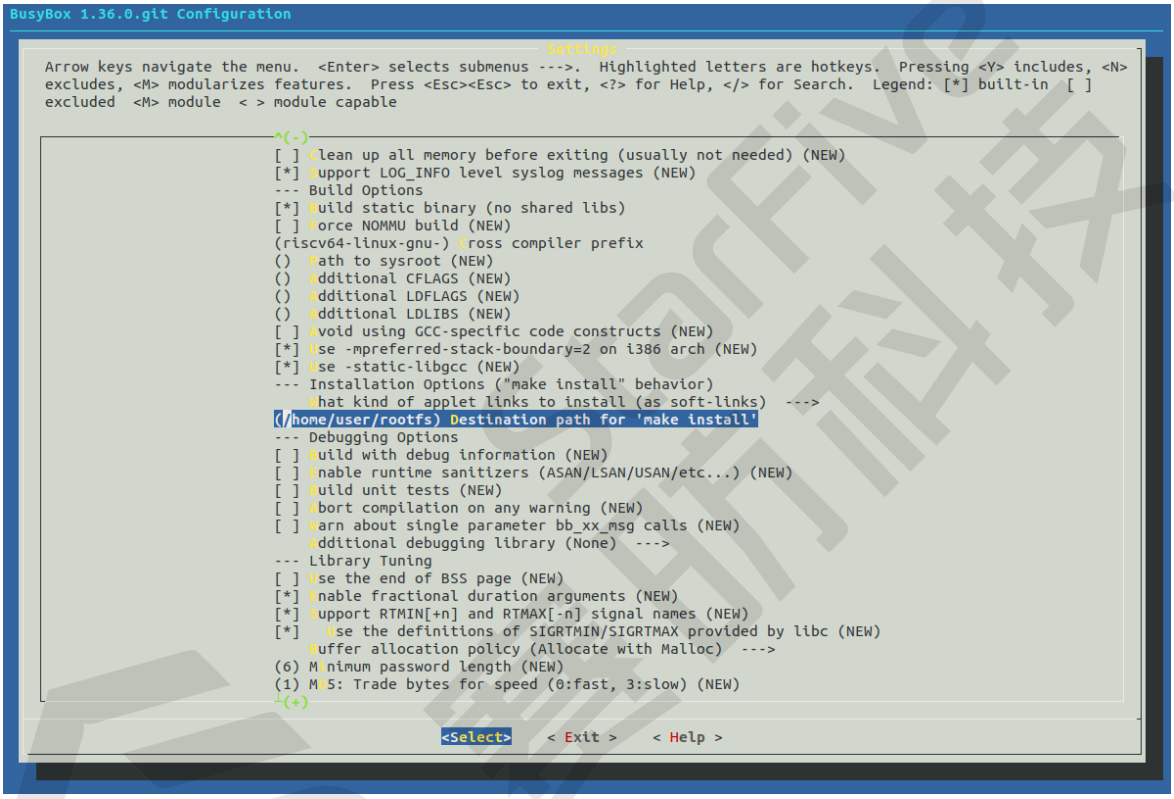
b. Type the following command:

```
riscv64-linux-gnu-
```

6. Under **Installation Options > >Destination path for 'make install'**, change the path to the path of the rootfs file directory (this is the installation location of the compiled BusyBox).
Example:

```
/home/user/rootfs
```

Figure 3-6 UI Example



7. Save the configuration and exit from the busybox configuration window.

8. Compile BusyBox.

```
make ARCH=riscv
```

9. Install BusyBox.

```
make install
```

10. Navigate to the `rootfs/etc` directory created before, create a file called `inittab` and open it using vim text editor.

```
cd rootfs/etc
vim inittab
```

11. Copy and paste the following content inside the `inittab` file.

```
::sysinit:/etc/init.d/rcS
::respawn:-/bin/login
::restart:/sbin/init
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r
::shutdown:/sbin/swapoff -a
```

12. Create a file called profile inside `rootfs/etc` and open it using vim text editor.

```
vim profile
```

13. Copy and paste the following content inside the profile file.

```
# /etc/profile: system-wide .profile file for the Bourne shells
echo
# echo -n "Processing /etc/profile... "
# no-op
# Set search library path
# echo "Set search library path in /etc/profile"
export LD_LIBRARY_PATH=/lib:/usr/lib
# Set user path
# echo "Set user path in /etc/profile"
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH
# Set PS1
# Note: In addition to the SHELL variable, ash supports \u, \h, \W, \$,
 \!, \n, \w, \nnn (octal numbers corresponding to ASCII characters)
# And \e[xx;xxm (color effects), etc.
# Also add an extra '\' in front of it!
# echo "Set PS1 in /etc/profile"
export PS1="\\e[00;32m[$USER@\\w\\a]\\$\\e[00;34m"
# echo "Done"
```

14. Create a file called `fstab` inside `rootfs/etc` and open it using vim text editor.

```
vim fstab
```

15. Copy and paste the following content inside the `fstab` file.

```
proc /proc proc defaults 0 0
none /tmp tmpfs defaults 0 0
mdev /dev tmpfs defaults 0 0
sysfs /sys sysfs defaults 0 0
```

16. Create a file called `passwd` inside `rootfs/etc` and open it using vim text editor.

```
vim passwd
```

17. Copy and paste the following content inside the `passwd` file.

```
root:x:0:0:root:/root:/bin/sh
```

18. Create a file called `group` inside `rootfs/etc` and open it using vim text editor.

```
vim group
```

19. Copy and paste the following content inside the group file.

```
root:x:0:root
```

20. Create a file called `shadow` inside `rootfs/etc` and open it using vim text editor.

```
vim shadow
```

21. Copy and paste the following content inside the `shadow` file.

```
root:BAy5qvelNWKns:1:0:99999:7:::
```

22. Create a directory called `init.d` inside `rootfs/etc` and navigate inside it.

```
mkdir init.d
cd init.d
```

23. Create a file called `rcS` inside `rootfs/etc/init.d` and open it using vim text editor.

```
vim rcS
```

24. Copy and paste the following content inside the `rcS` file.

```
#! /bin/sh
#echo "----------mount all"
/bin/mount -a
#echo "----------Starting mdev....."
#/bin/echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s
echo "*******************************************************"
echo " starfive mini RISC-V Rootfs"
echo "*******************************************************"
```

25. Navigate to the `rootfs/dev` directory created before and execute the following.

```
1 cd rootfs/dev
2 sudo mknod -m 666 console c 5 1
3 sudo mknod -m 666 null c 1 3
```

26. Create a soft link in the root directory of rootfs.

```
1 cd rootfs/
2 ln -s bin/busybox init
```

27. Modify the permissions of all files in the rootfs directory.

```
sudo chmod 777 -R *
```

28. Execute the following command in the rootfs directory to generate `rootfs.cpio.gz` (cpio file system package) in a different directory.

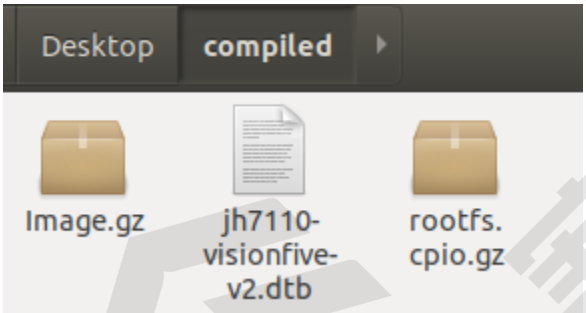
```
1 cd rootfs
2 find . | cpio -o -H newc | gzip > /home/user/Desktop/rootfs.cpio.gz
```

Note:
After you successfully run the command above, you will see a file named `rootfs.cpio.gz` on your Desktop. This directory can be any directory you want. If your CPU has 8 cores, change this to `-j8`. This process will take some time and therefore please wait patiently.

3.3. Moving Rootfs, Kernel, and dtb into VisionFive 2

Start by moving the previously compiled rootfs file system package, kernel and dtb images into a single directory.

Figure 3-7 Example Interface



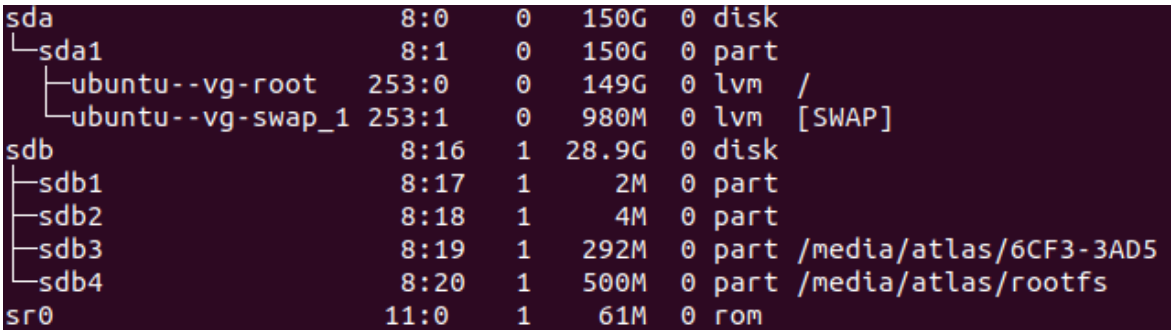
3.3.1. Method 1: Using Micro-SD Card

1. Insert a micro-SD card to the host PC.
2. Type the following to see the location of the connected micro-SD card.

```
lsblk
```

For example, it's `/dev/sdb`.

Figure 3-8 Example



3. Type the following to enter the partition configuration.

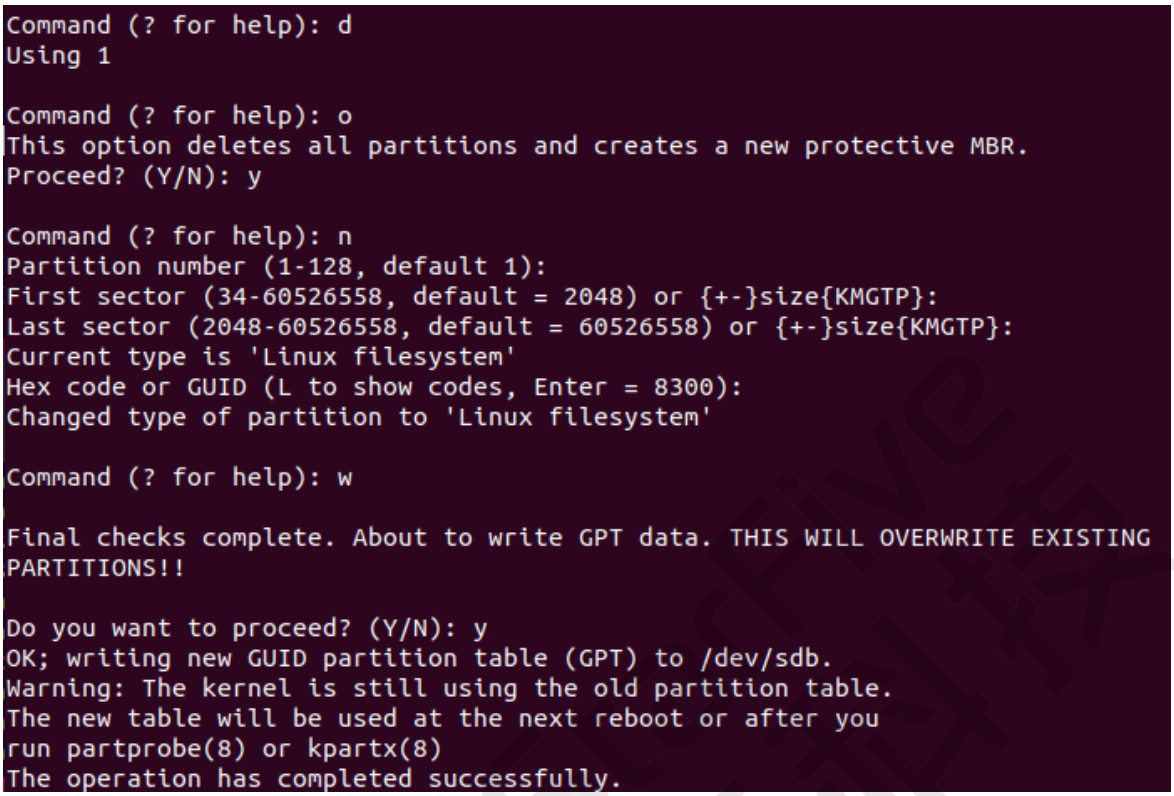
```
sudo gdisk /dev/sdb
```

Figure 3-9 Example Output

4. Delete the original partition and then create a new partition by entering the following respectively.

```
d--->o--->n--->w--->y
```

Figure 3-10 Example Command and Output



Tip:

Press `Enter` to keep some settings to default in this configuration.

5. Format the micro-SD card and create the file system.

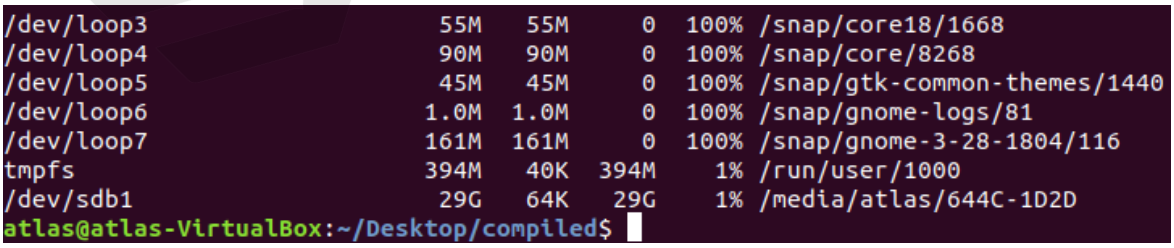
```
sudo mkfs.vfat /dev/sdb1
```

6. Remove the micro-SD card from PC and plug again to mount it.
7. Enter the following to check whether it gets mounted.

```
df -h
```

You will see an output as follows and take a note of the mount location.

Figure 3-11 Example Output



8. Navigate to the directory containing the 3 images as before.

```
cd Desktop/compiled
```

9. Copy the files to the micro-SD card by typing the following.

```
sudo cp Image.gz <Mount_Location>
sudo cp rootfs.cpio.gz <Mount_Location>
sudo cp <dtb_File_Name> <Mount_Location>
sync
```

Note:

- *<Mount_Location>*: the mount location as shown above.
- *<dtb_File_Name>*: the DTB file for VisionFive 2.

  Different boards use different dtb files:

  - `jh7110-visionfive-v2.dtb`: for Version 1.2A and 1.3B board.
  - `jh7110-visionfive-v2-ac108.dtb`: for version 1.2A and 1.3B board with ac108 codec.
  - `jh7110-visionfive-wm8960.dtb`: for Version 1.2A and 1.3B board with wm8960 codec.

Tip:
You can refer to the silk print on the board for version information.

Example:

The following are the example commands:

```
sudo cp Image.gz /media/user/644C-1D2D/
sudo cp rootfs.cpio.gz /media/user/644C-1D2D/
sudo cp jh7110-visionfive-v2.dtb /media/user/644C-1D2D/
sync
```

10. Remove the micro-SD card from PC, insert into VisionFive 2 and turn it on.
11. Open minicom while USB to Serial Adapter is connected between VisionFive 2 and PC, and wait until the board enters **U-Boot** mode. You will see the following output when it is in U-Boot mode.

Figure 3-12 Example Output

```
U-Boot 2021.10-00044-g135126c47b-dirty (Oct 28 2022 - 16:36:03 +0800)

CPU:   rv64imacu
Model: StarFive VisionFive V2
DRAM:  8 GiB
MMC:   sdio0@16010000: 0, sdio1@16020000: 1
```

12. Enter the following commands.

```
setenv kernel_comp_addr_r 0xb0000000;setenv kernel_comp_size
 0x10000000;
fatls mmc 1:1
fatload mmc 1:1 ${kernel_addr_r} Image.gz
fatload mmc 1:1 ${fdt_addr_r} jh7110-visionfive-v2.dtb
fatload mmc 1:1 ${ramdisk_addr_r} rootfs.cpio.gz
run chipa_set_linux;
booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Figure 3-13 Example Command and Output

```
StarFive # setenv kernel_comp_addr_r 0xb0000000;setenv kernel_comp_size 0x10000000;
StarFive # fatls mmc 1:1
            System Volume Information/
  7745113   Image.gz
    47743   jh7110-visionfive-v2.dtb
  1211720   rootfs.cpio.gz

3 file(s), 1 dir(s)

StarFive # fatload mmc 1:1 ${kernel_addr_r} Image.gz
7745113 bytes read in 330 ms (22.4 MiB/s)
StarFive # fatload mmc 1:1 ${fdt_addr_r} jh7110-visionfive-v2.dtb
47743 bytes read in 4 ms (11.4 MiB/s)
StarFive # fatload mmc 1:1 ${ramdisk_addr_r} rootfs.cpio.gz; run chipa_set_linux;
1211720 bytes read in 54 ms (21.4 MiB/s)
StarFive # booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
   Uncompressing Kernel Image
## Flattened Device Tree blob at 46000000
   Booting using the fdt blob at 0x46000000
   Using Device Tree in place at 0000000046000000, end 000000004600ea7e

Starting kernel ...
```

13. Log in by typing the following credentials.
    - Username: root
    - Password: starfive

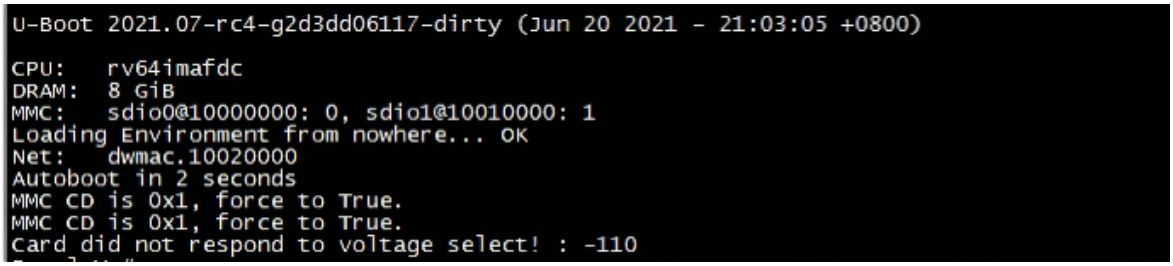
3.3.2. Method 2: Using Ethernet Cable

1. Connect an Ethernet Cable from the RJ45 port of VisionFive 2 to a router, connect serial adapter cable and power on the board.

   Note:
   Make sure the host PC is also connected to the same router using Ethernet or Wi-Fi.

2. Open **minicom** and wait until the board enters **U-Boot** mode. You will see the following output when it is in U-Boot mode.

Figure 3-14 Example Output



3. Enter the following commands to set U-Boot environment variables.

```
setenv serverip 192.168.125.142;setenv ipaddr 192.168.125.200;
setenv hostname starfive;setenv netdev eth0;
setenv kernel_comp_addr_r 0xb0000000;setenv kernel_comp_size
 0x10000000;
setenv bootargs console=ttyS0,115200 earlycon=sbi root=/dev/ram0
 stmmaceth=chain_mode:1 loglevel=8
```

Note:

Generally, the default IP of a router is 192.168.120.1. In this case, use the server IP as the IP assigned by the DHCP server of the router and use the VisionFive 2 IP as 192.168.120.xxx. However, if your router IP is different (e.g.: 192.168.2.1), the server and VisionFive 2 should follow the IP format of 192.168.2.xxx.

4. Check the connectivity by pinging the host PC from VisionFive 2.

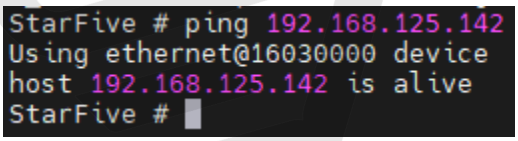
Example:

```
ping 192.168.120.12
```

Result:

If you see the following output, the host PC and VisionFive 2 have established communication on the same network.

Figure 3-15 Example Output



5. Install a TFTP server on the Host PC.

```
sudo apt-get update
sudo apt install tftpd-hpa
```

6. Check the status of the server.

```
sudo systemctl status tftpd-hpa
```

7. Execute the following to enter the TFTP server configuration.

```
sudo nano /etc/default/tftpd-hpa
```

8. Configure the TFTP server as follows.

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/home/user/Desktop/compiled"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

Note:
The `TFTP_DIRECTORY` is the directory that we created before with all the 3 images (`Image.gz`, `jh7110-visionfive-v2.dtb`, `rootfs.cpio.gz`).

9. Restart the TFTP server.

```
sudo systemctl restart tftpd-hpa
```

10. Type the following inside the U-Boot mode of VisionFive 2 to download the files from the TFTP server of the host PC and start the kernel.

```
tftpboot ${fdt_addr_r} <dtb_File_Name>;
tftpboot ${kernel_addr_r} Image.gz;tftpboot ${ramdisk_addr_r}
 rootfs.cpio.gz;
run chipa_set_linux;
booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Note:

Example:

The following command is an example for VisionFive 2:

```
tftpboot ${fdt_addr_r} jh7110-visionfive-v2.dtb;
tftpboot ${kernel_addr_r} Image.gz;
tftpboot ${ramdisk_addr_r} rootfs.cpio.gz;
run chipa_set_linux;
booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Result:

```
starfive mini RISC-V Rootfs
```

11. Log in with the following credentials.

    - Username: root
    - Password: starfive

4. Appendix

As mentioned earlier, the compiled Debian package, dtb, and kernel files can be transferred on VisionFive 2 through SCP network or USB drive. If there are no relevant devices (network cables, switches, USB drives), the SD card can be mounted on the system of the compilation host (requiring a card reader), and the relevant files can be directly copied to the partition of the SD card.

This chapter mainly introduces how to copy files to the corresponding partition of the SD card from the following three aspects.

4.1. Check Partition

Perform the following steps to check partition:

1. Insert a Micro-SD card with a burned Debian system (Debian202403, which was manually replaced with the kernel as mentioned earlier) into the compilation host, and execute the following command on the Ubuntu system to check the SD card partition:

```
$ lsblk
```

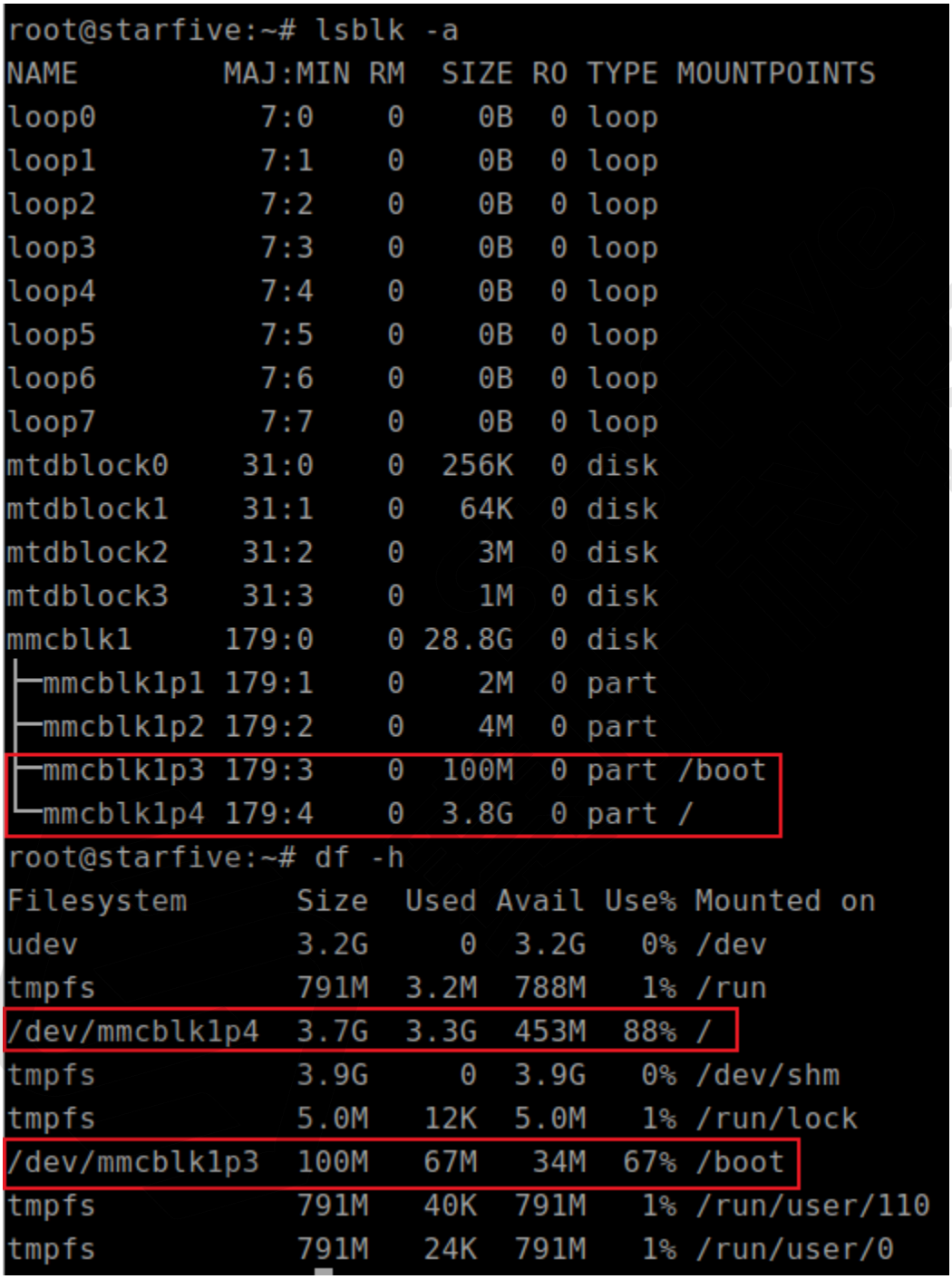
Figure 4-1 Example Output

```
➜  compile_kernel lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0         7:0    0 310.8M  1 loop /snap/code/156
loop1         7:1    0     4K  1 loop /snap/bare/5
loop2         7:2    0   311M  1 loop /snap/code/157
loop3         7:3    0  55.7M  1 loop /snap/core18/2812
loop4         7:4    0  63.9M  1 loop /snap/core20/2182
loop5         7:5    0  63.9M  1 loop /snap/core20/2264
loop6         7:6    0  74.1M  1 loop /snap/core22/1033
loop7         7:7    0  74.2M  1 loop /snap/core22/1122
loop9         7:9    0 164.8M  1 loop /snap/gnome-3-28-1804/198
loop10        7:10   0 269.6M  1 loop /snap/firefox/4136
loop11        7:11   0 400.8M  1 loop /snap/gnome-3-38-2004/112
loop12        7:12   0 349.7M  1 loop /snap/gnome-3-38-2004/143
loop13        7:13   0 504.2M  1 loop /snap/gnome-42-2204/172
loop14        7:14   0 505.1M  1 loop /snap/gnome-42-2204/176
loop15        7:15   0  91.7M  1 loop /snap/gtk-common-themes/1535
loop16        7:16   0  93.6M  1 loop /snap/p3x-onenote/220
loop17        7:17   0  12.9M  1 loop /snap/snap-store/1113
loop18        7:18   0  12.3M  1 loop /snap/snap-store/959
loop19        7:19   0  39.1M  1 loop /snap/snapd/21184
loop20        7:20   0  38.7M  1 loop /snap/snapd/21465
loop21        7:21   0   476K  1 loop /snap/snapd-desktop-integration/157
loop22        7:22   0   452K  1 loop /snap/snapd-desktop-integration/83
loop23        7:23   0  20.2M  1 loop /snap/v2raya/28
loop24        7:24   0  20.3M  1 loop /snap/v2raya/30
loop25        7:25   0 269.6M  1 loop /snap/firefox/4173
sda           8:0    0   1.8T  0 disk /run/timeshift/backup
                                      /coding
sdb           8:16   1  29.1G  0 disk
├─sdb1        8:17   1     2M  0 part
├─sdb2        8:18   1     4M  0 part
├─sdb3        8:19   1   100M  0 part
└─sdb4        8:20   1   3.8G  0 part
nvme0n1     259:0    0 476.9G  0 disk
├─nvme0n1p1 259:1    0   512M  0 part /boot/efi
└─nvme0n1p2 259:2    0 476.4G  0 part /var/snap/firefox/common/host-hunspell
```

As shown in the above figure, the system has read the SD card device and its partition information.

In Debian 202302 and 202403 image, the burned SD card will be divided into 4 partitions, with the 3rd and 4th partitions corresponding to the `/boot` and `/` partitions in the system respectively. We can verify this in the Debian system using commands such as `df` and `lsblk`:

Figure 4-2 Verification



4.2. Mount Partition

Execute the following command to create a path to mount partition, and mount the partition under that path:

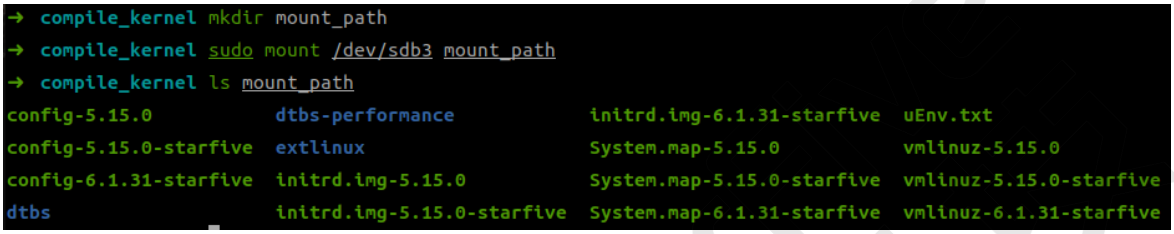
```
$ mkdir mount_path
```

- The 3rd path (/boot path):

  ```
  $ sudo mount /dev/sdb3 mount_path
  ```

  The following is the example output:

  Figure 4-3 Example Output

  Result: After mounting, files in the /boot path of the Debian system can be viewed in this path.

  Execute the following command to unmount partition:

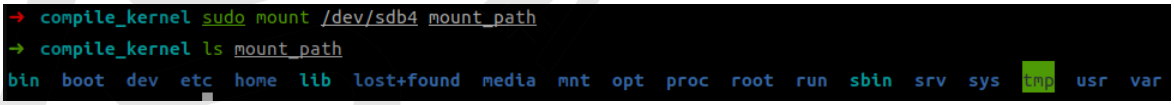
  ```
  $ sudo umount /deb/sdb3
  ```

- The 4th path (/ path):

  ```
  $ sudo mount /dev/sdb4 mount_path
  ```

  The following is the example output:

  Figure 4-4 Example Output

  Execute the following command to unmount partition:

  ```
  $ sudo umount /deb/sdb4
  ```

  Note:
  In embedded Linux systems, the /boot directory is usually used to store boot related files, such as boot loaders and kernel images, while the root directory / contains other files and directories of the system. The /boot directory and root directory / are usually divided into different partitions.

- When the `/boot` directory is in an independent partition, there is also a `/boot` directory in the root directory `/`.
- When the `/boot` partition is mounted to the root directory `/`, the `/boot` directory under the original root directory `/` will be hidden, and the content of the `/boot` partition will be exposed under the `/boot` path of the root directory `/`.

This approach separates bootloader and kernel image bootloader from the root file system. This can improve the security and stability of the system. However, it is necessary to avoid writing files in the `/boot` path of the root directory `/`, otherwise conflicts may arise when mounting on the `/boot` partition.

4.3. File Copying

After mounting the SD card partition, the required files can be copied to the target partition. In this section, taking Debian202403 image as an example, after mounting the 4th partition of the SD card, copy the files to the `/home/user` directory of the Debian system:

Note:
If you need to copy files to the 3rd and 4th partitions, please refer to Compile Kernel and Manually Replace Updated Files *(on page 30)* section.

1. Execute the following command to create test file:

```
$ echo "test message" > test_file.txt
```

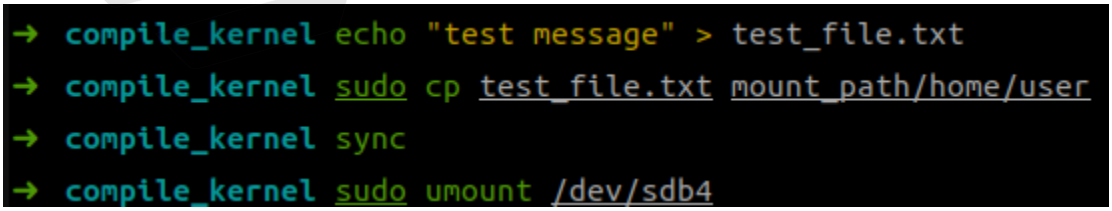
2. Execute the following command to copy the file to target path:

```
$ sudo cp test_file.txt mount_path/home/user && sync
```

3. Execute the following command to unmount partition:

```
$ sudo umount /dev/sdb4
```

Figure 4-5 File Copying



4. Insert the SD card on VisionFive 2, start and execute the following command to check:

```
ls /home/user
```

Figure 4-6 Example Output

```
root@starfive:~# ls /home/user/
test_file.txt
root@starfive:~# cat /home/user/test_file.txt
test message
```

Result: The file was correctly copied to the target path.