StarFive
赛昉科技

昉·星光 2单板计算机软件技术参考手册

版本：1.31

日期：2024/07/01

Doc ID: VisionFive 2-TRMCH-001

法律声明

阅读本文件前的重要法律告知。

版权注释

本文档中的说明均基于“视为正确”提供，可能包含部分错误。内容可能因产品开发而定期更新或修订。广东赛昉科技有限公司（以下简称“赛昉科技”）保留对本协议中的任何内容进行更改的权利，恕不另行通知。

赛昉科技明确否认任何形式的担保、解释和条件，无论是明示的还是默示的，包括但不限于适销性、特定用途适用性和非侵权的担保或条件。

赛昉科技无需承担因应用或使用任何产品或电路而产生的任何责任，并明确表示无需承担任何及所有连带责任，包括但不限于间接、偶然、特殊、惩戒性或由此造成的损害。

联系我们：

地址： 广东省佛山市顺德区大良街道云路社区昊阳路2号A区S201室

网站： http://www.starfivetech.com

邮箱：sales@starfivetech.com （销售） support@starfivetech.com（支持）

前言

关于本指南和技术支持信息

关于本手册

本手册主要讲解固件、U-Boot、Linux内核的编译方法，以及文件系统的制作方法。

Note:
赛昉科技为开发者提供了两种文档类型：网页版和PDF版。在执行命令时，为避免格式有误，请在网页版文档中复制命令。

修订历史

Table 0-1 修订历史

版本	发布说明	修订
1.31	2024/07/01	• 在软件环境 *(on page 15)*和编译内核、设备树与驱动模块 *(on page 27)*新增了一个步骤。 • 在更新配置文件 *(on page 30)*新增一个提示。 • 修改了创建SPL文件 *(on page 12)*、编译OpenSBI *(on page 13)*和创建fw_payload文件 *(on page 14)*命令中的小错误。
1.3	2024/05/11	新增了以下内容: • 编译并更新Linux内核 *(on page 15)*。 • 更换要加载的设备树文件 *(on page 32)*。 • 附录 *(on page 46)*
1.2	2023/05/17	更新了软件环境 *(on page 15)*。
1.1	2023/03/03	更新了创建SPL文件 *(on page 12)*中的步骤。
1.0	2022/12/26	首次发布。

注释和注意事项

本指南中可能会出现以下注释和注意事项:

- **Tip:**
  建议如何在某个主题或步骤中应用信息。
- **Note:**
  解释某个特例或阐释一个重要的点。
- **Important:**
  指出与某个主题或步骤有关的重要信息。

- **CAUTION:**
  表明某个操作或步骤可能会导致数据丢失、安全问题或性能问题。

- **Warning:**
  表明某个操作或步骤可能导致物理伤害或硬件损坏。

Contents

List of Tables

List of Figures

1. 硬件准备

请准备如下设备:

- 昉·星光 2
- 32 GB（或更大）的Micro SD卡
- 带有Linux/Windows/Mac操作系统的个人电脑
- USB转串口转换器
- 以太网电缆
- 电源适配器
- USB Type-C数据线
- 用于桌面环境使用:
  - 键盘和鼠标
  - 显示器或电视
  - HDMI电缆
- 此外，您可能还需要一些可选组件:
  - 以太网LAN电缆或兼容的WiFi dongle（默认启用ESWIN6600U或AIC8800模块）
  - USB转UART串行转换器模块

  Tip:
  用于通过UART启动模式进行系统恢复。

Note:
本手册中，PC主机安装的是Ubuntu 22.04 LTS。

2. 制作通用系统

本章介绍了如何制作通用系统。

主要包括以下部分：

2.1. 编译U-Boot和SPL

本章介绍了如何编译U-Boot和SPL。

主要包括以下部分：

2.1.1. 设置编译环境

请按照以下步骤设置您的交叉编译器：

1. 执行以下命令，安装Ubuntu软件包中的`riscv64-linux-gnu-gcc`编译器：

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

2. 执行以下命令检查`riscv64-linux-gnu-gcc`编译器的版本：

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

示例输出如下：

结果：

Figure 2-1 示例输出

```
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. 编译U-Boot

执行以下步骤，为昉·星光 2编译U-Boot：

1. 将U-Boot文件保存到您的目标目录下，如主目录（home directory）下：

```
cd ~ # home directory
```

2. 下载源代码，以编译U-Boot：

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

3. 执行以下命令，切换到代码分支：

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

4. 执行以下命令，在U-Boot目录下编译U-Boot：

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

Tip:

<Configuration_File>：在昉·星光 2上，该文件为`starfive_visionfive2_defconfig`。

结果：

编译完成后，在`u-boot`目录下将生成以下三个文件：

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

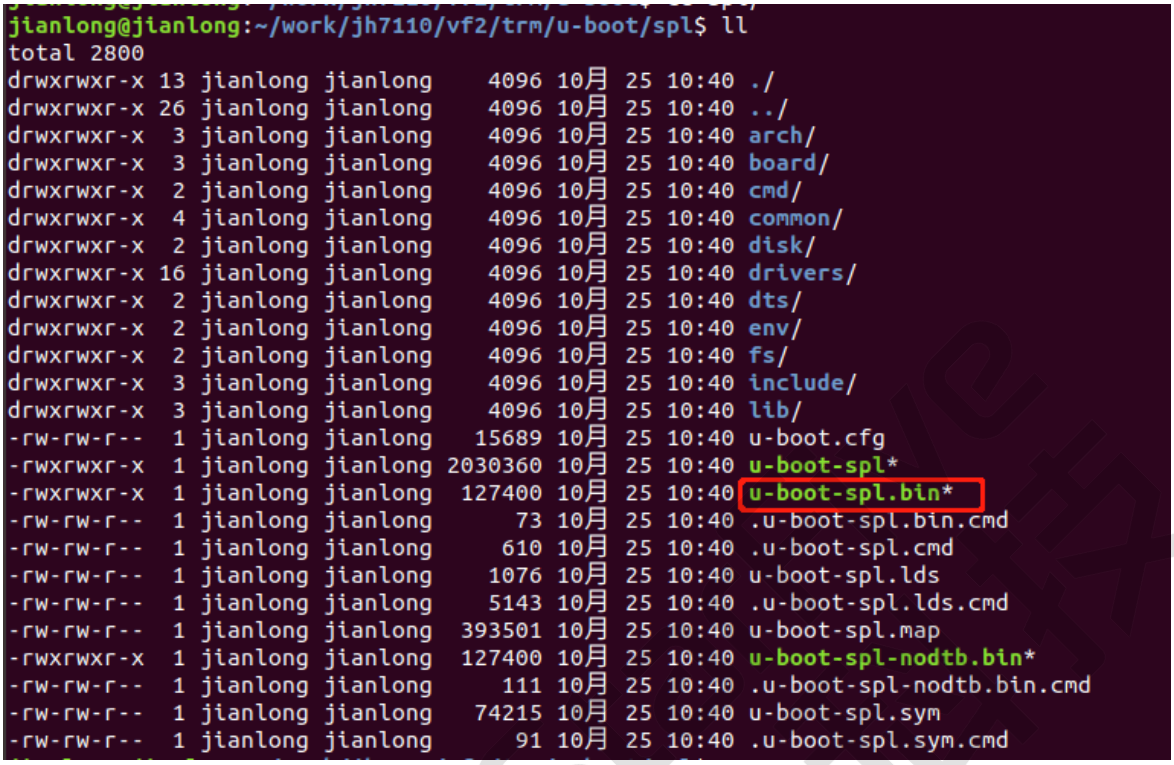
Figure 2-2 示例输出 - u-boot.bin

```
jianlong@jianlong:~/work/jh7110/vf2/trm/u-boot$ ll u-boot.bin
-rwxrwxr-x 1 jianlong jianlong 665952 10月 25 10:40 u-boot.bin*
```

Figure 2-3 示例输出 - visionfive2.dtb

```
jianlong@jianlong:~/work/jh7110/vf2/trm/u-boot$ ll arch/riscv/dts/starfive_visionfive2.dtb
-rw-rw-r-- 1 jianlong jianlong 39202 10月 25 10:40 arch/riscv/dts/starfive_visionfive2.dtb
```

Figure 2-4 示例输出 - u-boot-boot.bin



Tip:

- `starfive_visionfive2.dtb`和`u-boot.bin`都将用于稍后的OpenSBI编译。
- `u-boot-spl.bin`将用于稍后创建SPL文件。

2.1.3. 创建SPL文件

执行以下步骤，为昉·星光 2创建SPL文件：

1. 将工具文件保存到您的目标目录下，如主目录（`home directory`）下：

   命令示例：

   ```
   cd ~ # home directory
   ```

2. 下载源代码，以编译U-Boot：

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

3. 执行以下命令，切换到代码分支：

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

4. 执行以下命令，在`spl_tool`目录下创建SPL文件：

   ```
   cd spl_tool/
   make
   ```

Figure 2-5 示例输出

```
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. 执行以下命令生成SPL文件：

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

> **Tip:**
> 将*{U_BOOT_PATH}*修改为此前存放U-Boot文件的路径。

运行结果：

将生成名为`u-boot-spl.bin.normal.out`的文件。烧录`u-boot-spl.bin.normal.out`，请参见《昉·星光 2单板计算机快速参考手册》中“更新*SPL*和*U-Boot*”一节。

Figure 2-6 示例输出

```
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. 编译OpenSBI

OpenSBI全称为Open-source Supervisor Binary Interface，是开源Supervisor二进制接口，是一套RISC-V开源实现。它提供了RISC-V runtime服务，通常应用于ROM和LOADER后的启动阶段。典型的启动流程如下图所示：

Figure 2-7 典型启动流程

请参考以下步骤，为昉·星光 2编译OpenSBI：

1. 将OpenSBI文件保存到您的目标目录下，如主目录（`home directory`）下：

```
cd ~ # home directory
```

2. 下载源代码，以编译OpenSBI：

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

3. 在`opensbi`目录下，执行以下命令编译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:

将*{U_BOOT_PATH}*修改为此前存放U-Boot文件的路径

结果：编译完成后，在`opensbi/build/platform/generic/firmware`路径下，将生成大于2M的`fw_payload.bin`文件。

Figure 2-8 示例输出

```
jianlong@jianlong:~/work/jh7110/vf2/trm/opensbi/build/platform/generic/firmware$ ll
total 5544
drwxrwxr-x 3 jianlong jianlong    4096 10月 25 10:42 ./
drwxrwxr-x 6 jianlong jianlong    4096 10月 25 10:42 ../
-rwxrwxr-x 1 jianlong jianlong  152248 10月 25 10:42 fw_dynamic.bin*
-rw-rw-r-- 1 jianlong jianlong     792 10月 25 10:42 fw_dynamic.dep
-rwxrwxr-x 1 jianlong jianlong  979384 10月 25 10:42 fw_dynamic.elf*
-rw-rw-r-- 1 jianlong jianlong    1009 10月 25 10:42 fw_dynamic.elf.ld
-rw-rw-r-- 1 jianlong jianlong   76216 10月 25 10:42 fw_dynamic.o
-rwxrwxr-x 1 jianlong jianlong  152248 10月 25 10:42 fw_jump.bin*
-rw-rw-r-- 1 jianlong jianlong     712 10月 25 10:42 fw_jump.dep
-rwxrwxr-x 1 jianlong jianlong  978952 10月 25 10:42 fw_jump.elf*
-rw-rw-r-- 1 jianlong jianlong    1009 10月 25 10:42 fw_jump.elf.ld
-rw-rw-r-- 1 jianlong jianlong   72176 10月 25 10:42 fw_jump.o
-rwxrwxr-x 1 jianlong jianlong 2763112 10月 25 10:42 fw_payload.bin*
-rw-rw-r-- 1 jianlong jianlong     721 10月 25 10:42 fw_payload.dep
-rwxrwxr-x 1 jianlong jianlong 1645088 10月 25 10:42 fw_payload.elf*
-rw-rw-r-- 1 jianlong jianlong    1151 10月 25 10:42 fw_payload.elf.ld
-rw-rw-r-- 1 jianlong jianlong  738240 10月 25 10:42 fw_payload.o
drwxrwxr-x 2 jianlong jianlong    4096 10月 25 10:42 payloads/
```

2.1.5. 创建fw_payload文件

执行以下步骤，为昉·星光 2创建`fw_payload`:

1. 进入工具目录:

```
cd Tools/uboot_its
```

2. 复制编译OpenSBI的输出文件`fw_payload.bin`到目录路径下:

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

Note:
在执行命令前，将*{OPENSBI_PATH}*修改到OpenSBI的路径。

3. 执行以下命令，在`uboot_its`目录下创建`fw_payload`文件。

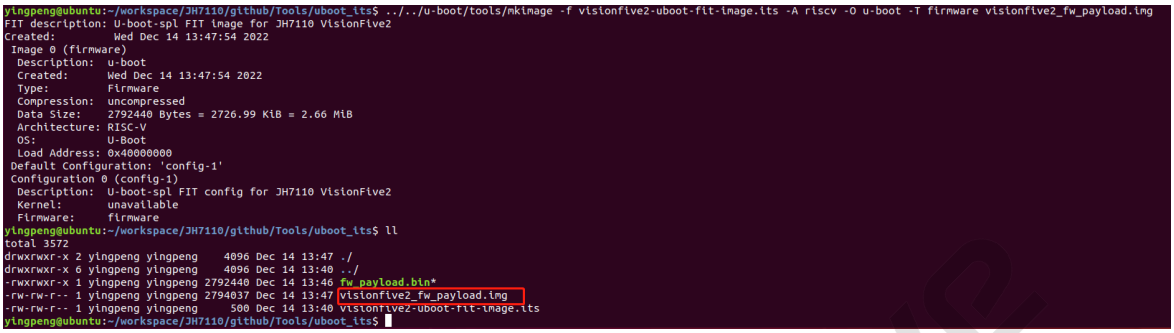
```
${U_BOOT_PATH}/tools/mkimage -f visionfive2-uboot-fit-image.its -A riscv -O u-boot -T firmware
 visionfive2_fw_payload.img
```

Note:
从PDF文档中复制此命令时，请删除换行符。

运行结果：

将生成名为visionfive2_fw_payload.img的文件。烧录visionfive2_fw_payload.img，请参见《昉·星光 2单板计算机快速参考手册》中“更新*SPL*和*U-Boot*”一节。

Figure 2-9 示例输出



2.2. 编译并更新Linux内核

本章主要介绍了以下几个方面：

2.2.1. 获取OS版本（以Debian OS为例）

步骤：

1. 访问此链接下载最新的操作系统。
2. 将最新版本的操作系统烧录到Micro-SD卡上。详细步骤请参考《昉·星光 2单板计算机快速参考手册》中的“将OS烧录到Micro-SD 卡上”章节。

2.2.2. 软件环境

按照以下步骤，搭建软件环境：

1. 执行以下命令，编译组件：

```
$ 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. 执行以下命令，下载源码：

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

3. 通过此链接查看Debian的发布信息，查找并将内核源码切换到对应版本，本节以Debian202403为例，对应的内核版本为v5.11.3。执行以下命令，切换分支：

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

下图为示例输出：

Figure 2-10 示例输出

4. 执行以下命令，设置编译Linux内核的默认设置：

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

> **Tip:**
> *<Configuration_File>*：在昉·星光 2上，该文件为`starfive_visionfive2_defconfig`。

5. （可选）修改配置文件。若需要修改内核配置，则执行以下命令：

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

以下为修改配置文件的一个示例：

将CPU默认调频策略从**ondemand**改为**performance**。

在**Devices Drivers > CPU Frequency scaling**下将**Default CPUFreq governor** 从**ondemand**改为**performance**并取消**ondemand**选项，如下图所示：

Figure 2-11 Ondemand

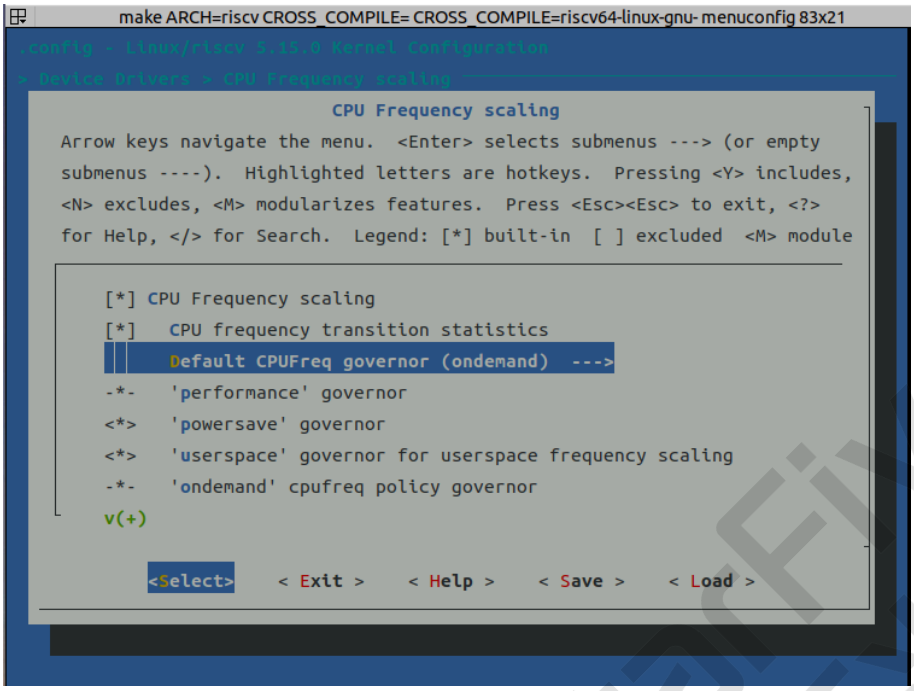
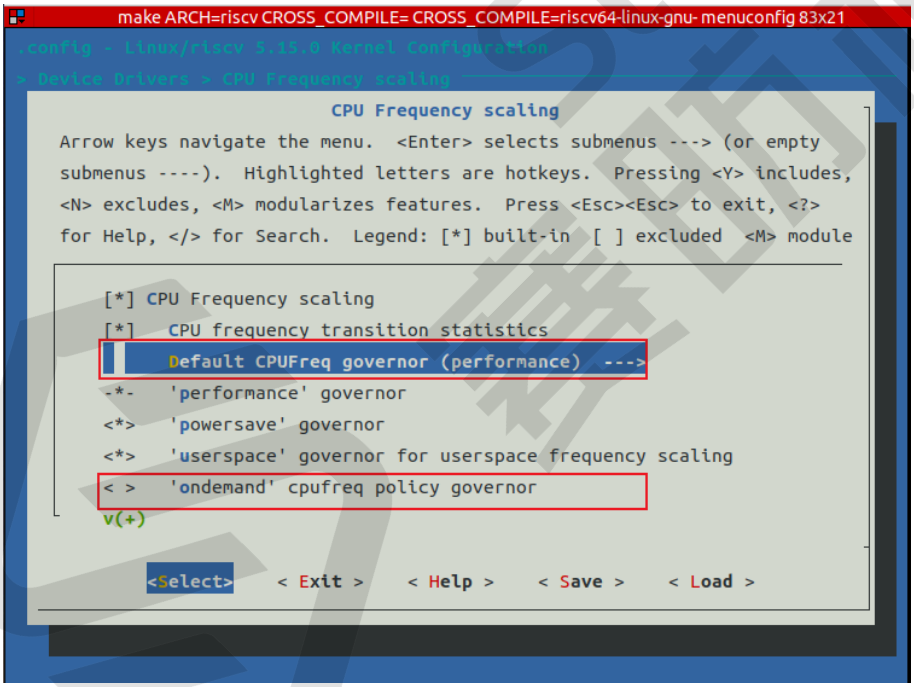


Figure 2-12 Performance



2.2.3. 编译Debian包并更新内核

执行软件环境 *(on page 15)*中前3步命令，并根据您的编译环境选择以下编译方法：

- 本地编译和安装 *(on page 18)*
- 交叉编译和安装 *(on page 18)*

本地编译和安装

1. 使用`bindeb-pkg`创建内核：

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

2. 编译完成后，安装`.deb`内核软件包。

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

交叉编译和安装

Tip:
可参考以下链接：

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

1. 执行以下命令，设置编译Linux内核的默认设置：

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

Tip:
<Configuration_File>：在昉·星光 2上，该文件为`starfive_visionfive2_defconfig`。

2. 通过以下命令，编译内核镜像以及头文件，并打包为Debian包：

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

Tip:
其中`local_version`为编译的内核的版本，在此例中设置为`-performance`，即执行：

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

3. 编译完成后，会在上一级生成以下文件和所需的设备树（`dtb`）文件：

Figure 2-13 文件

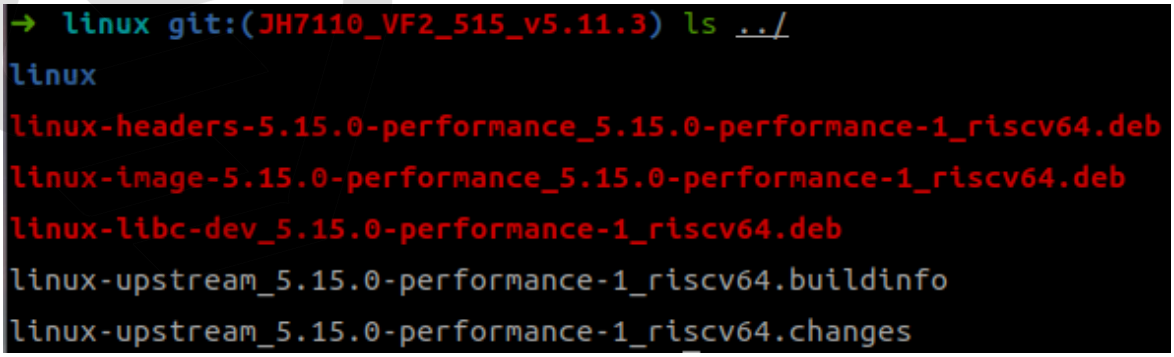


Figure 2-14 设备树文件

```
→ linux git:(JH7110_VF2_515_v5.11.3) ls arch/riscv/boot/dts/starfive
codecs                          jh7110-evb-pcie-i2s-sd.dts        jh7110-visionfive-v2-A10.dts
evb-overlay                     jh7110-evb-pinctrl.dtsi           jh7110-visionfive-v2-A11.dtb
jh7110-clk.dtsi                 jh7110-evb-spi-uart2.dtb          jh7110-visionfive-v2-A11.dts
jh7110-common.dtsi              jh7110-evb-spi-uart2.dts          jh7110-visionfive-v2-ac108.dtb
jh7110.dtsi                     jh7110-evb-uart1-rgb2hdmi.dtb     jh7110-visionfive-v2-ac108.dts
jh7110-evb-can-pdm-pwmdac.dtb   jh7110-evb-uart1-rgb2hdmi.dts     jh7110-visionfive-v2.dtb
jh7110-evb-can-pdm-pwmdac.dts   jh7110-evb-uart4-emmc-spdif.dtb   jh7110-visionfive-v2.dts
jh7110-evb.dtb                  jh7110-evb-uart4-emmc-spdif.dts   jh7110-visionfive-v2.dtsi
jh7110-evb.dts                  jh7110-evb-uart5-pwm-i2c-tdm.dtb  jh7110-visionfive-v2-sof-wm8960.dtb
jh7110-evb.dtsi                 jh7110-evb-uart5-pwm-i2c-tdm.dts  jh7110-visionfive-v2-sof-wm8960.dts
jh7110-evb-dvp-rgb2hdmi.dtb     jh7110-evb-usbdevice.dtb          jh7110-visionfive-v2-wm8960.dtb
jh7110-evb-dvp-rgb2hdmi.dts     jh7110-evb-usbdevice.dts          jh7110-visionfive-v2-wm8960.dts
jh7110-evb-i2s-ac108.dtb        jh7110-fpga.dtb                   Makefile
jh7110-evb-i2s-ac108.dts        jh7110-fpga.dts                   vf2-overlay
jh7110-evb-pcie-i2s-sd.dtb      jh7110-visionfive-v2-A10.dtb
```

4. 通过网络（SCP）或便携式存储介质（U盘）将编译生成的Debian包与设备树文件传到昉·星光 2上，下图为通过网络传输文件的示例输出：

Figure 2-15 示例输出

```
→ linux git:(JH7110_VF2_515_v5.11.3) cd ../
→ compile_kernel scp linux-* user@192.168.125.78:/home/user
The authenticity of host '192.168.125.78 (192.168.125.78)' can't be established.
ED25519 key fingerprint is SHA256:RXvDrZs5Z6dciIBroHX/g/18g4RryaudgjbIzIoLheQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.125.78' (ED25519) to the list of known hosts.
user@192.168.125.78's password:
linux-headers-5.15.0-performance_5.15.0-performance-1_riscv64.deb              100% 7362KB   8.1MB/s   00:00
linux-image-5.15.0-performance_5.15.0-performance-1_riscv64.deb                100%   11MB  31.1MB/s   00:00
linux-libc-dev_5.15.0-performance-1_riscv64.deb                                100% 1135KB  30.7MB/s   00:00
linux-upstream_5.15.0-performance-1_riscv64.buildinfo                          100% 6383     3.2MB/s   00:00
linux-upstream_5.15.0-performance-1_riscv64.changes                            100% 2177     2.6MB/s   00:00
```

5. 执行以下命令，安装Debian包：

```
$ 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. 安装完毕后，`/boot`下的文件更新为：

Figure 2-16 /boot下文件

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

2.2.3.1. 更新配置文件

昉·星光 2的Debian镜像在过去的一段时间里，内部启动机制经过几次修改（主要体现在dtb文件加载地址），需要对不同版本的Debian镜像的内核配置更新进行说明：

- Debian202403 *(on page 20)*
- Debian202302 - Debian202311 *(on page 25)*

2.2.3.1.1. Debian202403

更新配置文件

按照以下步骤，更新Debian202403版本镜像的配置文件：

1. 在安装编译生成的文件之前，/boot路径下的extlinux/extlinux.conf、uEnv.txt以及dtbs/ 路径的目录结构如下所示：

Figure 2-17 目录结构

```
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 目录结构

```
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 目录结构

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

6 directories, 0 files
```

2. 在安装过后，/boot路径下新增以下几个文件：

Figure 2-20 新增文件

```
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
```

同时，extlinux/extlinux.conf文件被修改，见下图：

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. 可见各个启动项中的`fdtdir`都被设为了`/dtbs`，结合`uEnv.txt`中的配置可知，昉·星光 2上电后将会按照`/boot/dtbs/starfive/jh7110-visionfive-v2.dtb`的路径加载设备树文件，显然这与`dtbs/`路径下的目录结构不符合。因此，`label l0`、`label l0r`、`label l1`、`label l1r`启动选项的`fdtdir`应修改为未安装Debian包前的状态：

Figure 2-23 修改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. 执行以下命令，为新的内核启动选项设置新的dtb寻址路径：

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

并将编译的内核源码下的设备树文件放到此路径下：

Figure 2-24 放置内核源码

```
root@starfive:/boot# tree ./dtbs-performance/
./dtbs-performance/
└── 5.15.0
    └── starfive
        └── jh7110-visionfive-v2.dtb

2 directories, 1 file
```

5. 修改`extlinux/extlinux.conf`中`label l2`与`label l2r`中的`fdtdir`配置，使得启动新内核时，会从`/boot/dtbs-performance/5.15.0/starfive/jh7110-visionfive-v2.dtb`路径加载设备树文件：

Figure 2-25 修改fdtdir配置

```
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-performance/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 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-performance/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
```

验证

替换Debian202403镜像的内核并上电后，在U-Boot menu选择新增的内核选项，如下面两张图分别选择了1和5的内核启动选项，即可看到正确加载了各自对应的设备树文件：

Figure 2-26 内核启动选项-1

```
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
6:      Debian GNU/Linux bookworm/sid 5.15.0-performance (rescue target)
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 409 ms (21.6 MiB/s)
Retrieving file: /vmlinuz-6.1.31-starfive
8985236 bytes read in 397 ms (21.6 MiB/s)
append: root=/dev/mmcblk1p4 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.dtb
49157 bytes read in 10 ms (4.7 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 000000004600f004

Starting kernel ...
```

Figure 2-27 内核启动选项-5

```
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
6:      Debian GNU/Linux bookworm/sid 5.15.0-performance (rescue target)
Enter choice: 5
5:      Debian GNU/Linux bookworm/sid 5.15.0-performance
Retrieving file: /initrd.img-5.15.0-performance
9203350 bytes read in 407 ms (21.6 MiB/s)
Retrieving file: /vmlinuz-5.15.0-performance
8432978 bytes read in 373 ms (21.6 MiB/s)
append: root=/dev/mmcblk1p4 root=/dev/mmcblk1p4 rw console=tty0 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:
1 selinux=0
Retrieving file: /dtbs-performance/5.15.0/starfive/jh7110-visionfive-v2.dtb
52430 bytes read in 10 ms (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 ...
```

上电登录后，输入以下命令，查看系统信息：

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

Figure 2-28 系统信息

```
user@starfive:~$ cat /proc/version
Linux version 5.15.0-performance (atlas@atlas-ThinkStation-P350) (riscv64-linux-gnu-gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.
0, GNU ld (GNU Binutils for Ubuntu) 2.38) #1 SMP Tue Apr 23 11:40:16 CST 2024
```

2.2.3.1.2. Debian202302 - Debian202311

Debian202302到Debian202311中的启动配置均相同，通过Debian包替换内核也较为简单，本节以Debian202311为例：

更新配置文件

按照以下步骤，更新Debian202311版本镜像的配置文件：

1. 编译并安装内核Debian包后，创建放置`dtb`文件的路径：
2. 执行以下命令，为新的内核启动选项设置新的`dtb`寻址路径：

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

   并将编译的内核源码下的设备树文件放到此路径下：

   Figure 2-29 放置内核源码

   ```
   root@starfive:/boot# tree ./dtbs-performance/
   ./dtbs-performance/
   └── starfive
       └── jh7110-visionfive-v2.dtb

   1 directory, 1 file
   ```

3. 修改`extlinux/extlinux.conf`文件:

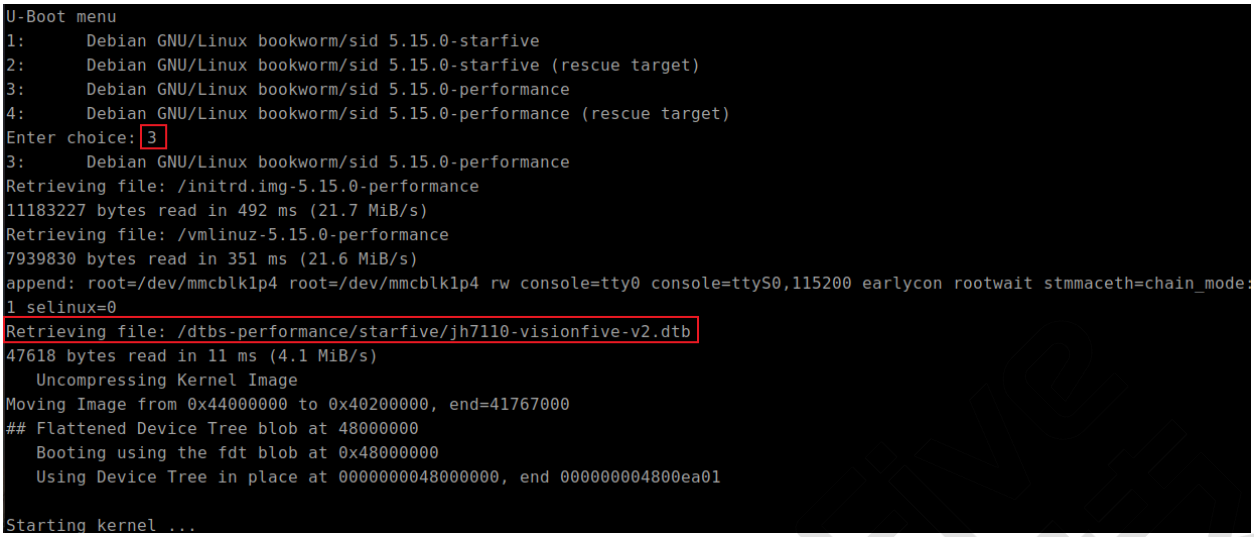
Figure 2-30 修改文件

```
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
```

验证

上电启动后，在U-Boot menu选择新增的内核选项，如下图选择了3的内核启动选项，即可看到正确加载了对应的设备树文件:

Figure 2-31 内核启动选项-3



2.2.4. 编译内核并手动替换更新文件

本节主要介绍了以下两个部分：

- 编译内核、设备树与驱动模块 *(on page 27)*
- 更换要加载的设备树文件 *(on page 32)*

2.2.4.1. 编译内核、设备树与驱动模块

按照以下步骤，编译内核、设备树与驱动模块：

1. 执行以下命令，设置编译Linux内核的默认设置：

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

> **Tip:**
> *<Configuration_File>*：在昉·星光 2上，该文件为`starfive_visionfive2_defconfig`。

2. 在执行软件环境的设置后，执行以下命令编译源码：

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

3. 执行以下命令，建立一个存放生成内核文件的目录：

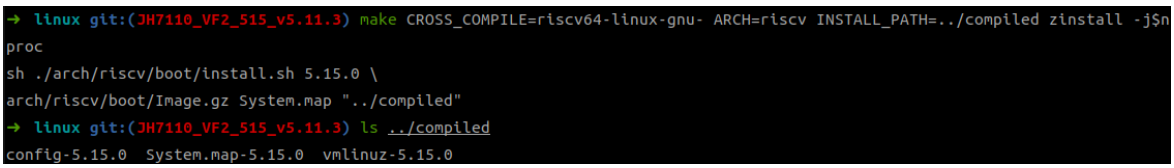
```
$ mkdir ../compiled
```

4. 执行以下命令，编译生成`config`，`System.map`，`vmlinuz`这几个文件到指定路径下：

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

下图为示例输出：

Figure 2-32 示例输出



5. 输入以下命令，复制`dtb`文件：

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

6. （可选）执行以下命令，编译生成模块文件并安装到指定路径下：

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

Note:
若新的内核不涉及驱动模块的改动，此项可不执行，仍使用默认内核的驱动模块。

下图为示例输出：

Figure 2-33 示例输出

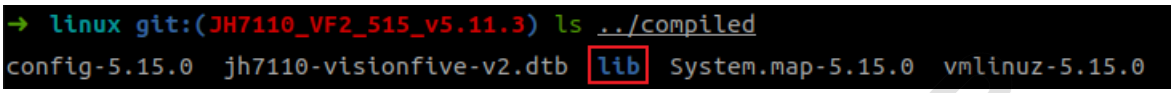
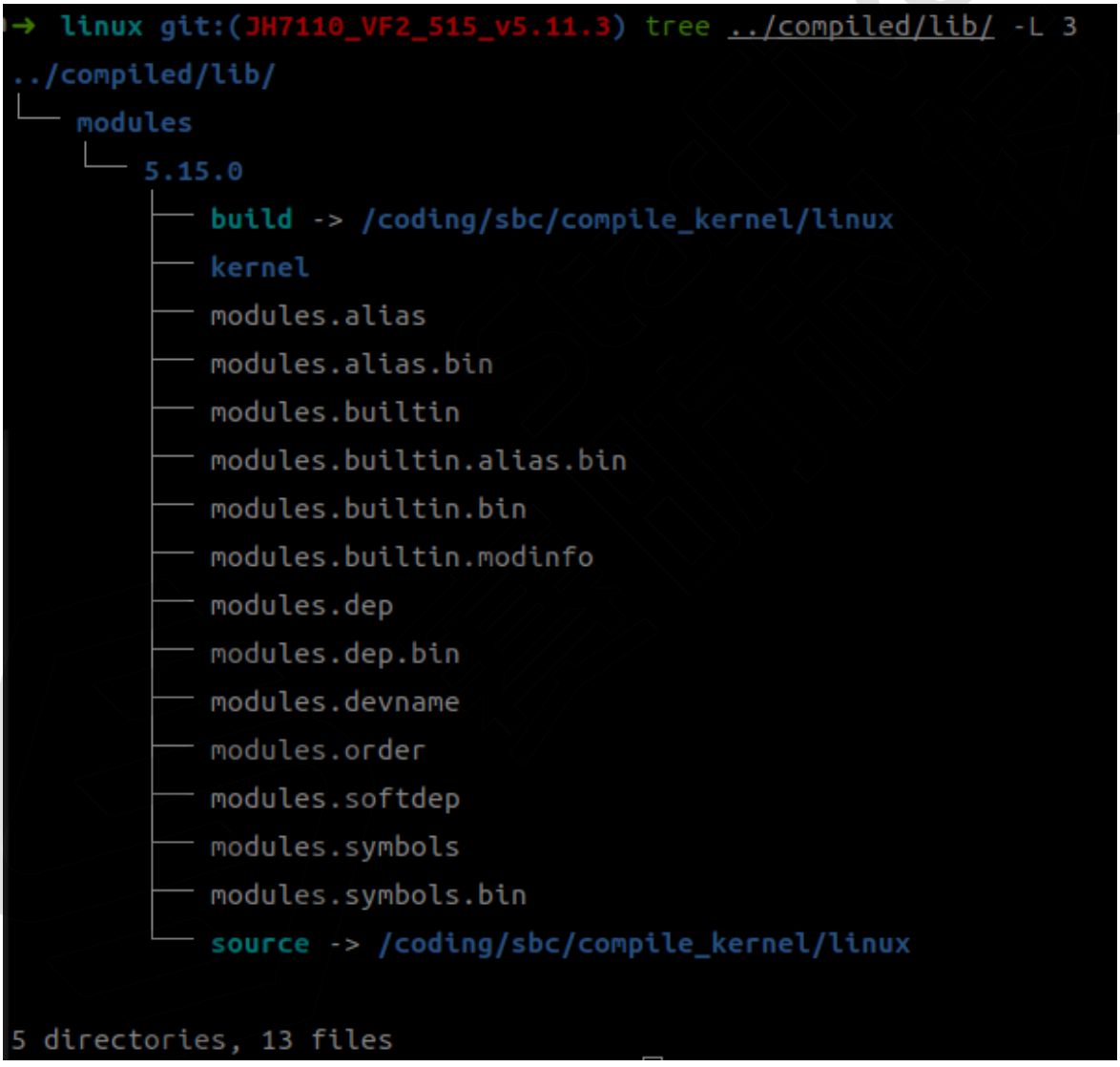


Figure 2-34 示例输出



Note:
Debian镜像的对应路径下并无build与source的链接，上图中的对应项可删除。

2.2.4.2. 替换内核文件并更新配置文件

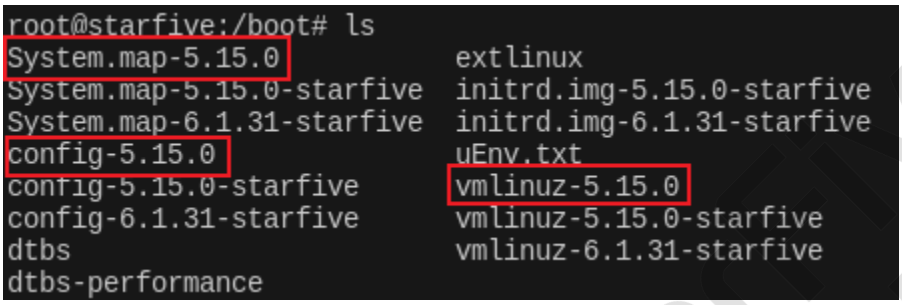
本节主要介绍了以下两个内容：

2.2.4.2.1. 替换内核文件

将compiled/路径下编译生成的文件通过网络或可移动存储介质放到运行Debian的昉·星光 2上，并将各个文件放到对应路径下。

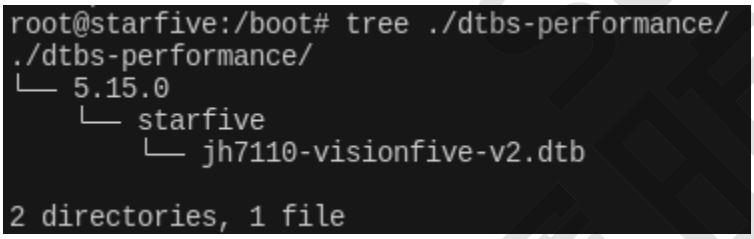
1. 将System.map-5.15.0、config-5.15.0与vmlinuz-5.15.0放置到/boot路径下:

   Figure 2-35 放置文件

2. 设备树文件参考默认路径，创建与版本对应的新路径（此例中为/boot/dtbs-performance/5.15.0/starfive）并将jh7110-visionfive-v2.dtb放置其中:

   Figure 2-36 放置文件

3. （可选）将lib/modules下的文件放到昉·星光 2上的/lib/modules路径下:

   Figure 2-37 放置文件

4. （可选）进入对应版本的内核模块路径，执行以下命令生成initramfs:

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

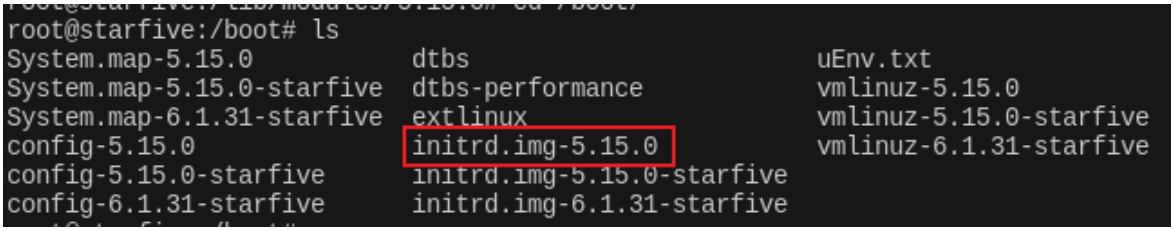
   Figure 2-38 生成initramfs

   结果:

   在/boot路径下生成对应版本号的initrd.img文件。

Figure 2-39 initrd.img



Note:

`update-initramfs`是一个用于生成`initramfs`（初始内存文件系统）的命令。`-xxx`这个参数指定了要为哪个内核版本生成`initramfs`。您需要将`-xxx`替换为您要生成`initramfs`的实际内核版本号（此例中为`5.15.0`）。

2.2.4.2.2. 更新配置文件

替换上述文件后，需要修改`extlinux/extlinux.conf`文件以增加新内核的启动项：

Figure 2-40 修改extlinux/extlinux.conf文件

```
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
```

验证

按照以下步骤进行验证：

1. 重新上电后，可在U-Boot menu查看到新增的启动选项，选择对应选项后，可见initrd.img、vmlinux以及dtb等文件均正确的从设定路径加载。

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 版本

```
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. 此外，前面提到，若新的内核不涉及驱动模块的改动，可不执行modules_install部分命令并生成替换对应版本的initrd.img文件，这样也可正常启动内核。

   下面修改extlinux.conf文件中新增启动项，将initrd配置由生成的initrd.img-5.15.0改为默认的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:**
> 关于fdtdir的设置，请参考更新配置文件 *(on page 20)*。

3. 重新上电后并在U-Boot menu选择对应选项，可见加载了initrd.img-5.15.0-starfive，且系统正常启动：

Figure 2-44 U-Boot Menu

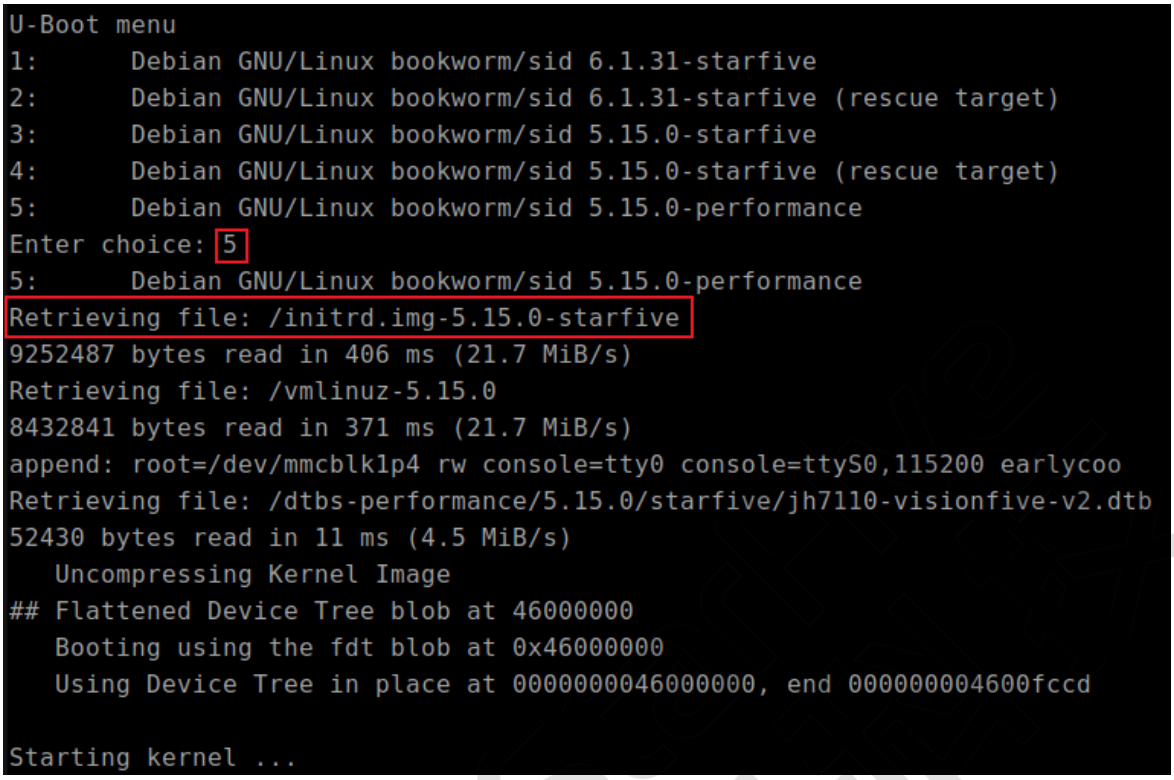
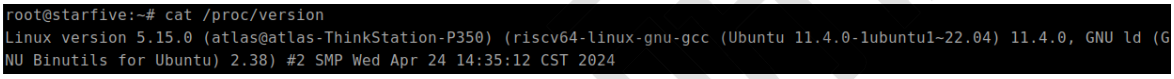


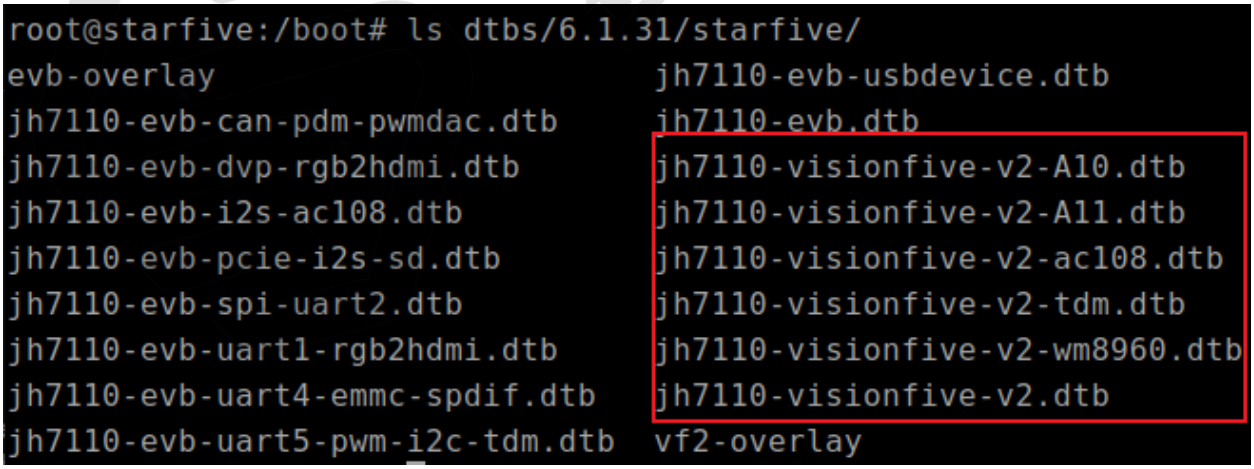
Figure 2-45 版本



2.3. 更换要加载的设备树文件

通过启用不同的设备树文件，可使得昉·星光 2实现不同的功能或支持不同的外设。以Debian202403的默认内核为例，其支持以下不同的设备树文件：

Figure 2-46 设备树文件



Note:
不同的开发板使用不同的dtb文件：

- `jh7110-visionfive-v2.dtb`：用于1.2A和1.3B版的开发板。
- `jh7110-visionfive-v2-ac108.dtb`：用于带有ac108编解码器的1.2A和1.3B版的开发版。
- `jh7110-visionfive-v2-tdm.dtb`：用于带有tdm声卡的1.2A和1.3B版的开发版
- `jh7110-visionfive-v2-wm8960.dtb`：用于带有wm8960编解码器的1.2A和1.3B版的开发板。

修改uEnv.txt文件

通过修改`uEnv.txt`文件，可使板子启动时加载不同的设备树文件。例如，修改`uEnv.txt`文件，使其使用`jh7110-visionfive-v2-wm8960.dtb`以支持wm8960编解码器：

Figure 2-47 修改uEnv.txt文件

```
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
```

验证

重新上电并选择对应内核后，可以发现`jh7110-visionfive-v2-wm8960.dtb`已从对应的路径正确加载：

Figure 2-48 验证

```
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. 制作BusyBox系统

本节介绍了如何制作BusyBox系统。

主要包括以下部分：

3.1. 编译Linux（交叉编译）

按照以下步骤，交叉编译Linux：

1. 执行以下命令，安装依赖包以创建内核：

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

2. 从赛昉科技Github仓库取内核文件：

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

3. 执行以下命令，切换到代码分支：

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

4. 执行以下命令，设置编译Linux内核的默认设置：

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

> **Tip:**
> *<Configuration_File>*：在昉·星光 2上，该文件为`starfive_visionfive2_defconfig`。

5. 执行以下命令，设置编译Linux内核其他软件设置：

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

6. 编译Linux内核：

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

> **Note:**
> 按照CPU内核的数量，调整此命令*-jx*的值。如果您的CPU中有8个内核，请将其更改为*-j8*。该过程较为耗时，请您耐心等待。

结果：

系统将在`linux/arch/riscv/boot`目录下，生成内核镜像文件`Image.gz`。

Figure 3-1 示例输出

```
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$
```

系统将在`linux/arch/riscv/boot`目录下，生成dtb文件文件Image.gz。

Figure 3-2 生成dtb文件

```
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
```

在稍后移植rootfs、dtb和内核到昉·星光 2上时，将使用到`Image.gz`和`.dtb`文件。

3.2. 制作文件系统

执行以下步骤，制作文件系统：

1. 创建目录结构：

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

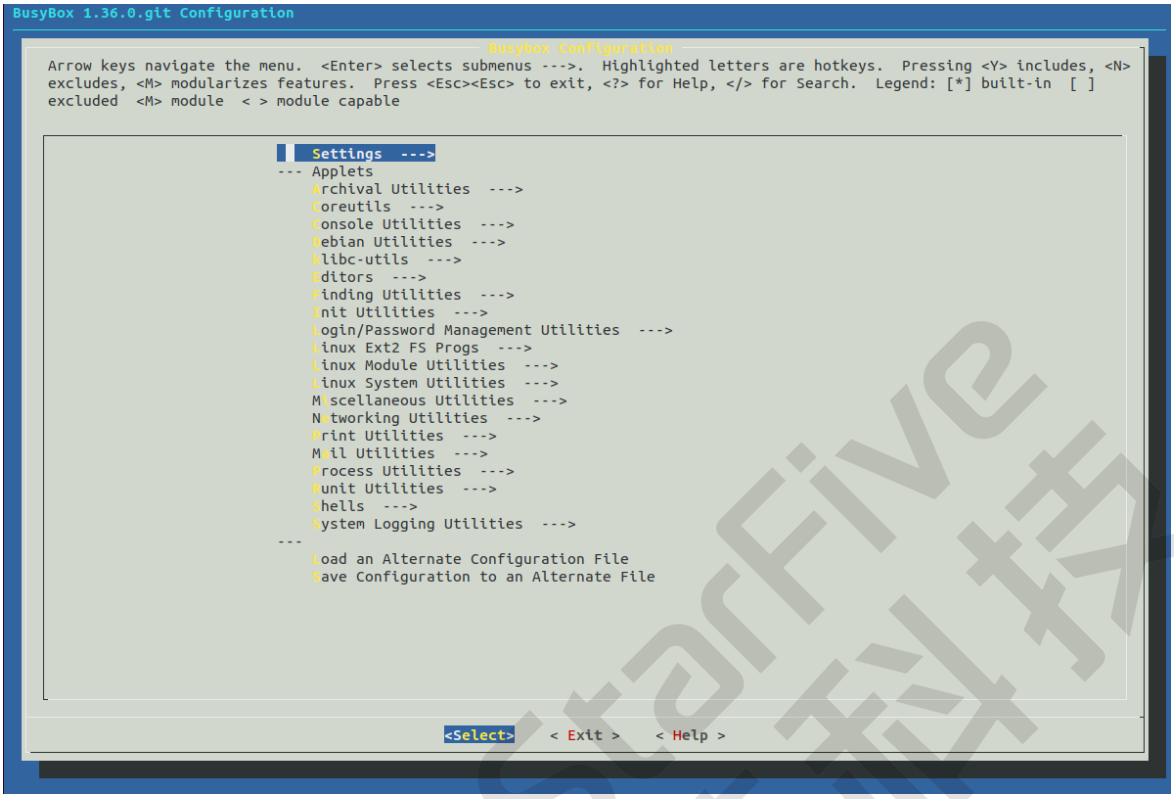
2. 下载BusyBox源代码，保存至rootfs文件夹以外的路径：

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

3. 找到解压后文件所在位置，并进入BusyBox配置界面：

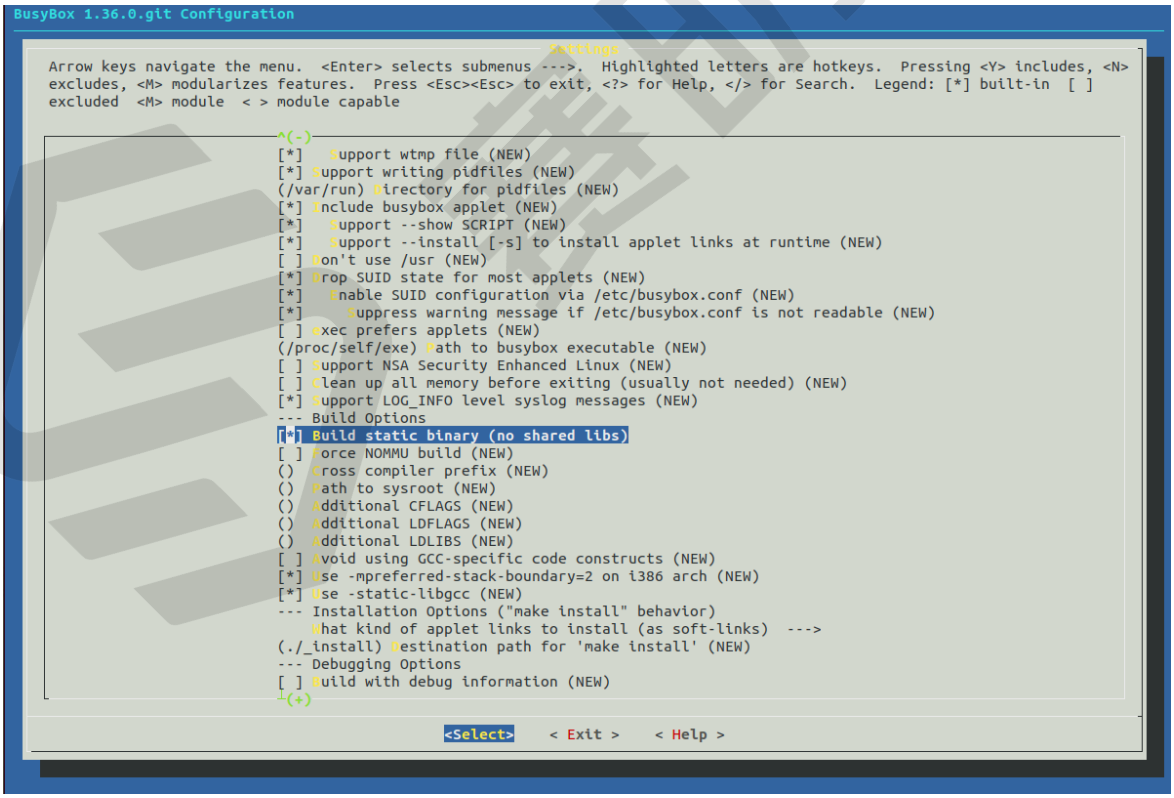
```
cd busybox
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv menuconfig
```

Figure 3-3 配置Busybox



4. 选择Settings > Build Options，按Y检查Build static binary (no shared libs)选项。

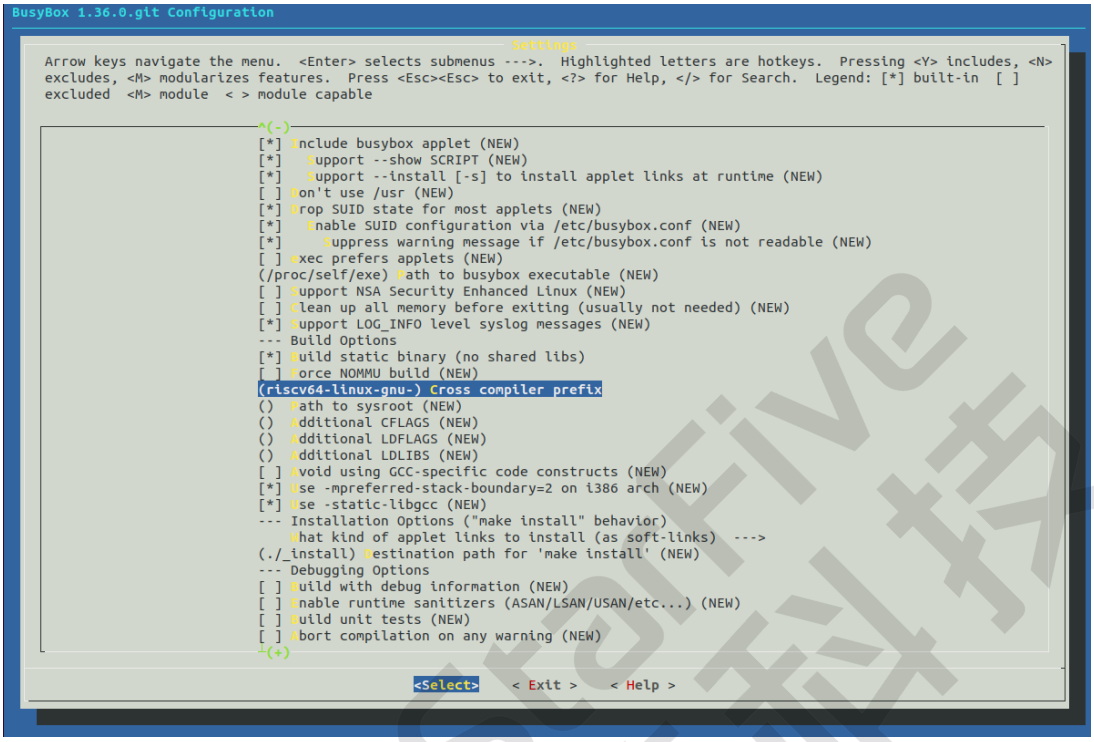
Figure 3-4 检查Build static binary (no shared libs)



5. 指定编译器。

a. 在**Build Options**下，选择**(riscv64-linux-gnu-) Cross compiler prefix**。

Figure 3-5 选择Cross Compiler Prefix



b. 执行以下命令指定编译器：

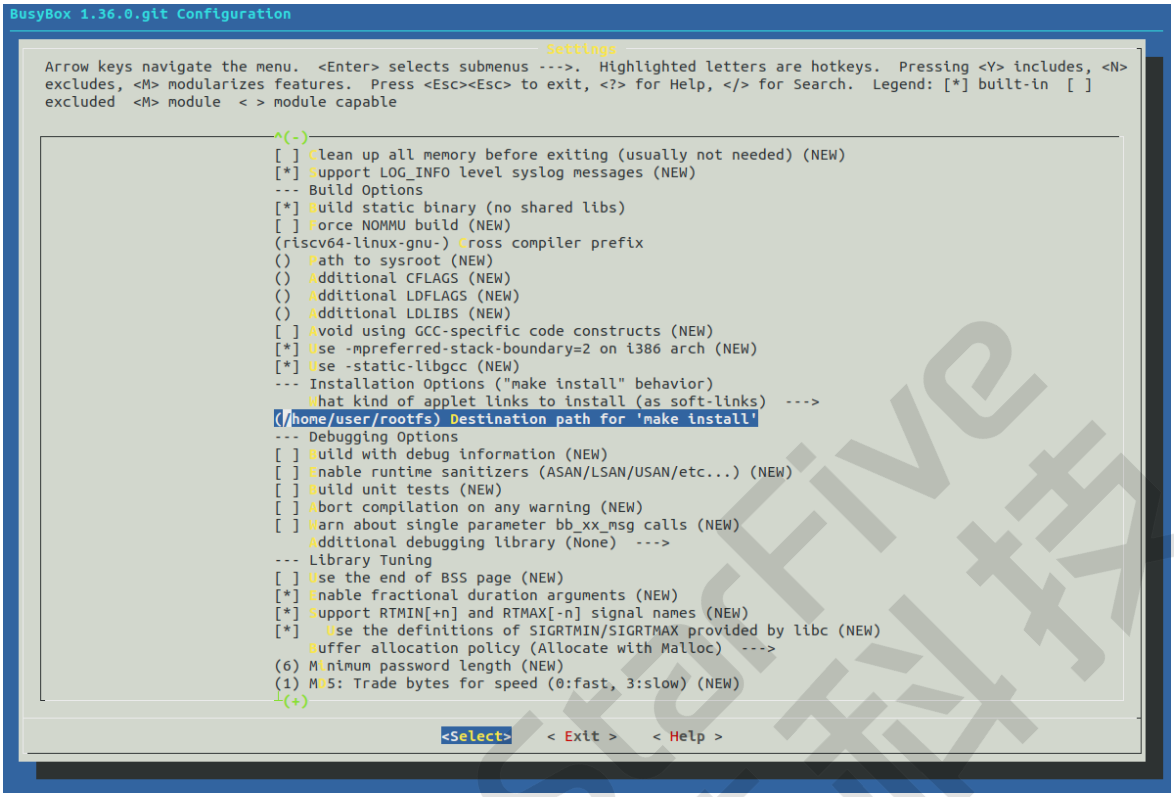
```
riscv64-linux-gnu-
```

6. 选择**Installation Options > Destination path for 'make install'**下，将路径更改为rootfs文件目录的路径（即编译后的BusyBox的安装路径）。

   示例：

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

Figure 3-6 示例界面



7. 保存设置内容，退出BusyBox设置窗口。
8. 编译BusyBox：

```
make ARCH=riscv
```

9. 安装BusyBox：

```
make install
```

10. 进入此前创建的`rootfs/etc`目录，创建一个名为`inittab`的文件，并使用vim文本编辑器打开。

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

11. 复制以下内容，并粘贴到`inittab`文件内。

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

12. 在`rootfs/etc`目录下，新建名为`profile`的文件，并使用vim文本编辑器打开。

```
vim profile
```

13. 复制以下内容，并粘贴到`profile`文件内。

```
# /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. 在`rootfs/etc`目录下，新建名为`fstab`的文件，并使用vim文本编辑器打开。

```
vim fstab
```

15. 复制以下内容，并粘贴到`fstab`文件内。

```
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. 在`rootfs/etc`目录下，新建名为`passwd`的文件，并使用vim文本编辑器打开。

```
vim passwd
```

17. 复制以下内容，并粘贴到`passwd`文件内。

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

18. 在`rootfs/etc`目录下，新建名为`group`的文件，并使用vim文本编辑器打开。

```
vim group
```

19. 复制以下内容，并粘贴到`group`文件内。

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

20. 在`rootfs/etc`目录下，新建名为`shadow`的文件，并使用vim文本编辑器打开。

```
vim shadow
```

21. 复制以下内容，并粘贴到`shadow`文件内。

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

22. 在`rootfs/etc`目录下，新建名为`init.d`的目录，并到该目录下。

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

23. 在`rootfs/etc`目录下，新建名为`rcS`的文件，并使用vim文本编辑器打开。

```
vim rcS
```

24. 复制以下内容，并粘贴到`rcS`文件内。

```
#! /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. 进入此前创建的`rootfs/dev`目录，并执行以下操作:

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

26. 在rootfs的根目录下新建软链接：

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

27. 修改rootfs目录中所有文件的权限：

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

28. 在rootfs目录下，执行以下命令在指定目录下生成`rootfs.cpio.gz`（cpio文件系统包）。

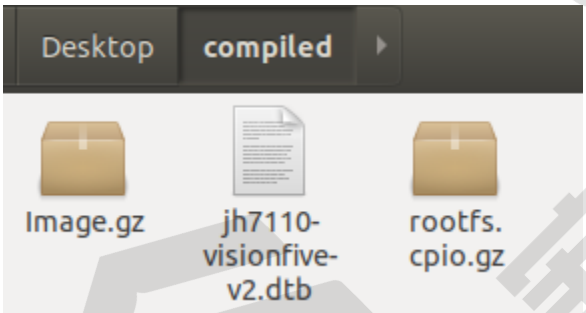
```
1 cd rootfs
2 find .| cpio -o -H newc | gzip > /home/user/Desktop/rootfs.cpio.gz
```

Note:
系统成功执行命令后，将在桌面上生成名为`rootfs.cpio.gz`的文件。您也可以根据需要，将命令中的目录修改为其它路径。如果您的CPU中有8个内核，请将其更改为`-j8`。该过程较为耗时，请您耐心等待。

3.3. 移植Rootfs，内核和dtb到昉·星光 2

首先，我们需要将此前编译的rootfs系统软件包、内核和dtb镜像文件移动到同一目录下。

Figure 3-7 示例界面



3.3.1. 方法1：使用Micro SD卡

1. 将Micro-SD卡插入计算机；
2. 输入以下命令，查看连接中的Micro-SD卡地址：

```
lsblk
```

如下图所示，示例中的Micro-SD卡地址为`/dev/sdb`。

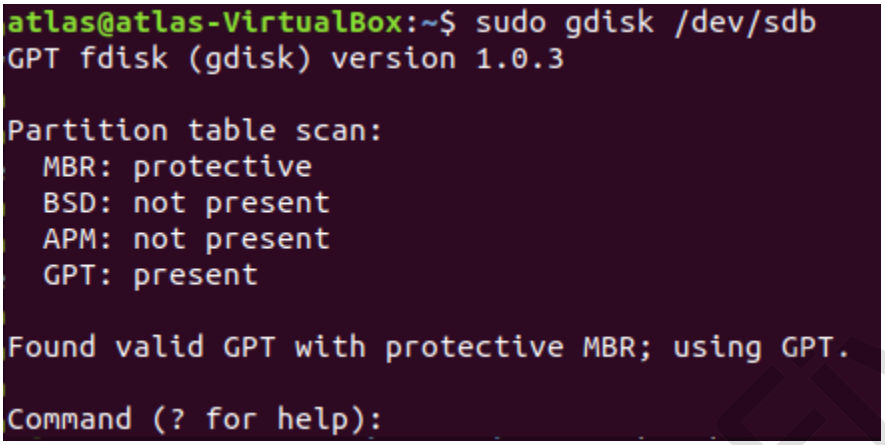
Figure 3-8 示例

```
sda                      8:0    0   150G  0 disk
└─sda1                   8:1    0   150G  0 part
  ├─ubuntu--vg-root    253:0    0   149G  0 lvm  /
  └─ubuntu--vg-swap_1  253:1    0   980M  0 lvm  [SWAP]
sdb                      8:16   1  28.9G  0 disk
├─sdb1                   8:17   1     2M  0 part
├─sdb2                   8:18   1     4M  0 part
├─sdb3                   8:19   1   292M  0 part /media/atlas/6CF3-3AD5
└─sdb4                   8:20   1   500M  0 part /media/atlas/rootfs
sr0                     11:0    1    61M  0 rom
```

3. 输入以下命令，进入分区配置：

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

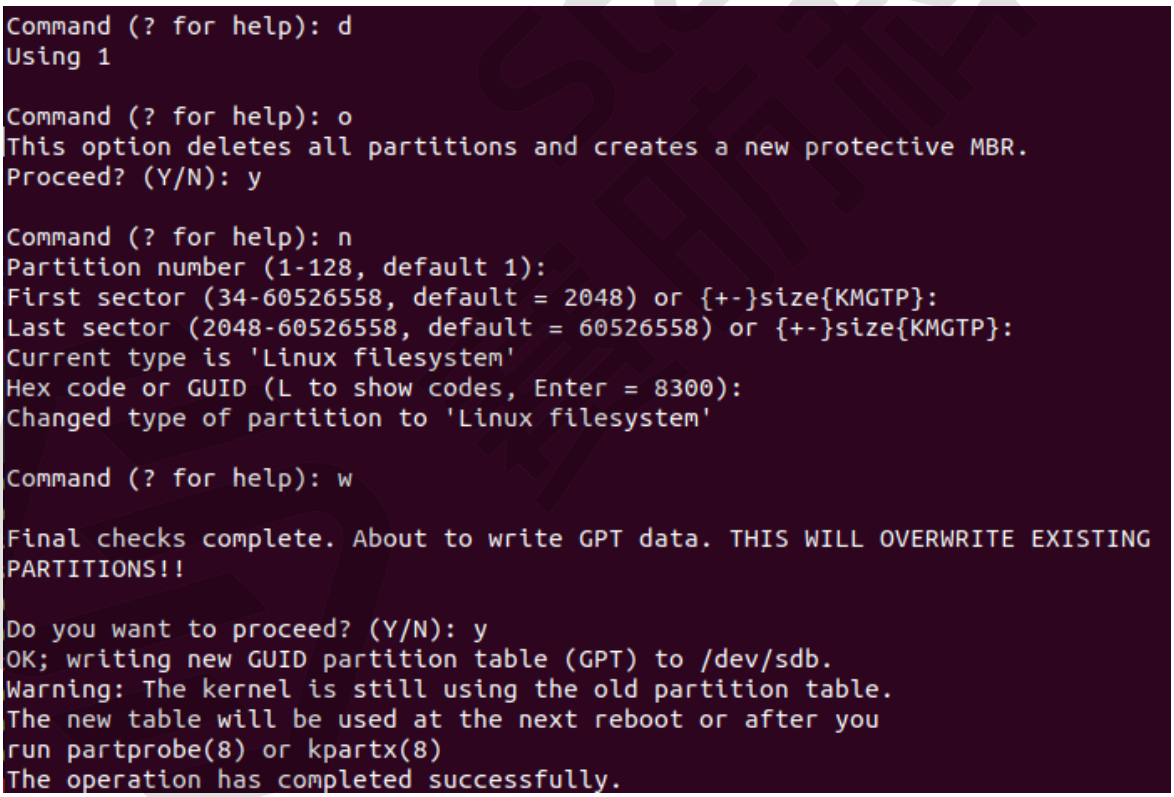
Figure 3-9 示例输出



4. 分别输入以下命令，删除原来的分区并创建新的分区：

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

Figure 3-10 示例命令和输出



Tip:
为保持某些默认设置，请按Enter回车键。

5. 格式化Micro-SD卡，并创建文件系统：

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

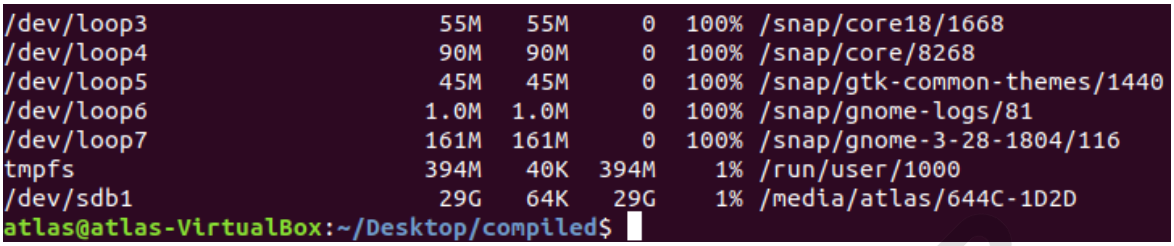
6. 从计算机中移除Micro-SD卡，并重新插入以挂载系统镜像。

7. 输入以下命令查看是否挂载成功：

```
df -h
```

系统输出如下，请记录下图高亮处的挂载路径。

Figure 3-11 示例输出



8. 进入到rootfs系统软件包、内核和dtb这三个镜像文件所在路径：

```
cd Desktop/compiled
```

9. 输入以下命令复制镜像文件到Micro-SD 卡：

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

Note:

- *<Mount_Location>*：此前记录的挂载路径。
- *<dtb_File_Name>*：昉·星光 2的DTB文件。

  不同的开发板使用不同的dtb文件：

  - `jh7110-visionfive-v2.dtb`：用于1.2A和1.3B版的开发板。
  - `jh7110-visionfive-v2-ac108.dtb`：用于带有ac108编解码器的1.2A和1.3B版的开发版。
  - `jh7110-visionfive-v2-wm8960.dtb`：用于带有wm8960编解码器的1.2A和1.3B版的开发板。

Tip:
您可查看开发板上的丝印获取版本信息。

示例：

命令示例：

```
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. 从计算机中移除Micro SD卡，并将该卡插入昉·星光 2，然后启动。
11. 使用USB转串口转换器，将昉·星光 2连接至计算机，然后打开minicom，等待昉·星光 2进入U-Boot模式。以下示例输出表明昉·星光 2已进入U-Boot模式：

Figure 3-12 示例输出

```
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. 输入以下命令：

```
setenv kernel_comp_addr_r 0xb0000000;
setenv kernel_comp_size 0x10000000;setenv kernel_addr_r 0x44000000;
setenv fdt_addr_r 0x48000000;setenv ramdisk_addr_r 0x48300000
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
booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

Figure 3-13 示例命令和输出

```
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. 输入以下用户名和密码登录：
    - Username: root
    - Password: starfive

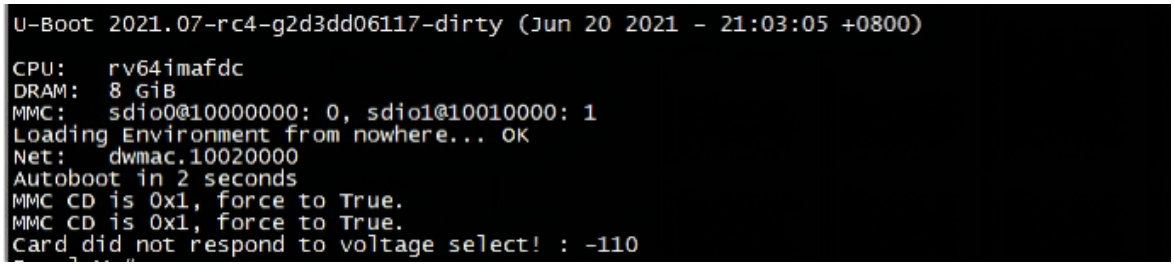
3.3.2. 方法2：使用网线

1. 使用网线通过昉·星光 2上的RJ45接口和路由器连接，连接串口转换器，然后启动开发板。

> **Note:**
> 请确保主机PC也通过网络或Wi-Fi与路由器连接。

2. 打开minicom，等待开发板进入U-Boot模式。以下示例输出表明昉·星光 2已进入U-Boot模式：

Figure 3-14 示例输出



3. 执行以下命令，设置U-Boot的环境变量：

```
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 kernel_addr_r 0x44000000;
setenv fdt_addr_r 0x48000000;setenv ramdisk_addr_r 0x48300000;
setenv bootargs console=ttyS0,115200 earlycon=sbi root=/dev/ram0 stmmaceth=chain_mode:1 loglevel=8
```

Note:
一般情况下路由器的默认IP为192.168.120.1。在这种情况下，请使用由路由器的DHCP服务器分配的IP，昉·星光 2的IP地址应为192.168.120.xxx。但是，如果您的路由器IP地址不同（如：192.168.2.1），请确保服务器IP和昉·星光 2属于同一IP段（例如192.168.2.xxx）中。

4. 通过ping命令，检查主机PC与昉·星光 2的连接情况。

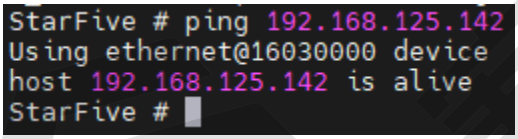
示例：

```
ping 192.168.120.12
```

结果：

以下输出表明主机PC与昉·星光 2已经在同一网络下建立连接。

Figure 3-15 示例输出



5. 在主机PC上安装TFTP服务器：

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

6. 检查服务器状态：

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

7. 输入以下命令进入TFTP服务器配置：

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

8. 执行以下命令设置TFTP服务器：

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

Note:
`TFTP_DIRECTORY`是之前我们创建的目录，包含三个镜像文件（`Image.gz`，`jh7110-visionfive-v2.dtb`和`rootfs.cpio.gz`）

9. 重启TFTP服务器：

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

10. 在昉·星光 2的U-Boot模式下输入以下命令，从主机PC的TFTP服务器下载文件，并启动内核：

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

Note:

示例：

以下命令是昉·星光 2的一个示例：

```
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}
```

结果：

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

11. 输入以下用户名和密码登录：
    - Username: root
    - Password: starfive

4. 附录

前面提到，为了将编译生成的Debian包、dtb以及内核等文件传输到昉·星光 2上，可以通过SCP网络传输或挂载U盘等方式。若没有相关设备（网线、交换机、U盘），还可通过将SD卡挂载到编译主机的系统下（需要读卡器），直接将相关文件拷贝到SD卡的分区中。

本章主要从以下三个方面介绍了如何将文件拷贝到SD卡对应分区。

4.1. 查看分区

按照以下步骤，查看分区：

1. 将带有已烧录Debian系统（此处为上文中手动替换过内核的Debian202403）的Micro-SD卡插入插入编译主机，在Ubuntu系统下执行以下命令，检查SD卡分区：

```
$ lsblk
```

Figure 4-1 示例输出

```
➜  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
```

由上图可知，系统已读取到SD卡设备以及其分区信息。

在Debian202302-Debian202403中，烧录的SD卡都会被划分为4个分区，其中的第3个与第4个分区分别对应系统下的/boot以及/分区，我们可在Debian系统下通过df与lsblk等命令验证这一点：

Figure 4-2 验证

```
root@starfive:~# lsblk -a
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0         7:0    0    0B  0 loop
loop1         7:1    0    0B  0 loop
loop2         7:2    0    0B  0 loop
loop3         7:3    0    0B  0 loop
loop4         7:4    0    0B  0 loop
loop5         7:5    0    0B  0 loop
loop6         7:6    0    0B  0 loop
loop7         7:7    0    0B  0 loop
mtdblock0    31:0    0  256K  0 disk
mtdblock1    31:1    0   64K  0 disk
mtdblock2    31:2    0    3M  0 disk
mtdblock3    31:3    0    1M  0 disk
mmcblk1     179:0    0 28.8G  0 disk
├─mmcblk1p1 179:1    0    2M  0 part
├─mmcblk1p2 179:2    0    4M  0 part
├─mmcblk1p3 179:3    0  100M  0 part /boot
└─mmcblk1p4 179:4    0  3.8G  0 part /
root@starfive:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            3.2G     0  3.2G   0% /dev
tmpfs           791M  3.2M  788M   1% /run
/dev/mmcblk1p4  3.7G  3.3G  453M  88% /
tmpfs           3.9G     0  3.9G   0% /dev/shm
tmpfs           5.0M   12K  5.0M   1% /run/lock
/dev/mmcblk1p3  100M   67M   34M  67% /boot
tmpfs           791M   40K  791M   1% /run/user/110
tmpfs           791M   24K  791M   1% /run/user/0
```

4.2. 挂载分区

执行以下命令，创建一个路径用于挂载分区，将想要查看的分区挂载在该路径下：

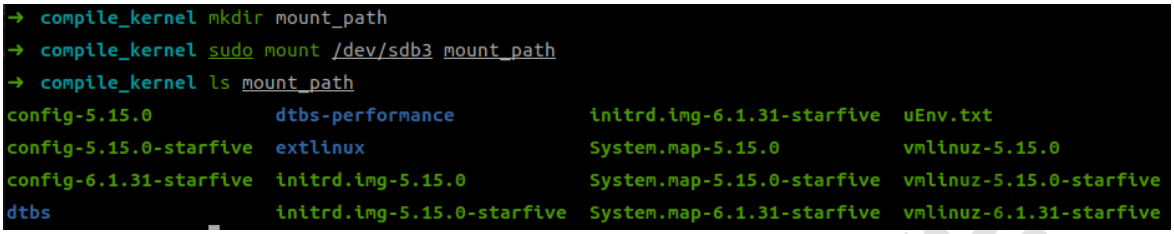
```
$ mkdir mount_path
```

- 第3分区（/boot路径）：

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

以下为示例输出：

Figure 4-3 示例输出



结果：挂载后可在此路径下查看Debian 系统中/boot路径下的文件。

如需卸载分区，可执行以下命令：

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

- 第4分区（/路径）：

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

以下为示例输出：

Figure 4-4 示例输出

如需卸载分区，可执行以下命令：

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

Note:
在嵌入式Linux系统中，通常会将/boot目录用于存储引导加载程序和内核映像等引导相关的文件，而根目录/则包含了系统的其他文件和目录。/boot目录与根目录/通常会划分到不同的分区。

- 当/boot目录在独立分区时，根目录/中也有一个/boot目录。
- 当/boot分区被挂载到根目录/时，原本根目录/下的/boot目录会被隐藏，而/boot分区中的内容会被暴露在根目录/的/boot路径下。

这种方式将引导加载程序和内核镜像等引导程序独立于根文件系统。这样可以提高系统的安全性和稳定性。但是，需要避免在根目录/的/boot路径下写入文件，否则会在/boot分区挂载时产生冲突。

4.3. 文件拷贝

挂载SD卡分区后，可将所需文件拷贝到目标分区中，本节以Debian202403为例，在挂载SD卡的第4分区后，将文件拷贝到Debian系统的/home/user目录下：

Note:
如需拷贝文件到第3分区和第4分区，请参考编译内核并手动替换更新文件 *(on page 27)*一节。

1. 执行以下命令，新建测试文件：

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

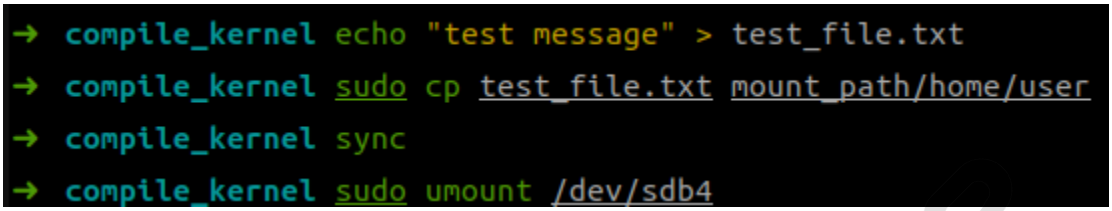
2. 执行以下命令，拷贝文件到目标路径：

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

3. 执行以下命令，卸载分区：

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

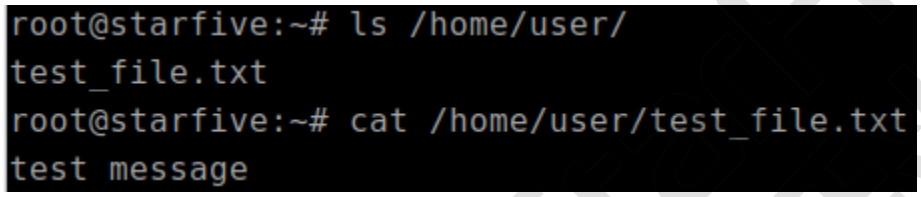
Figure 4-5 文件拷贝



4. 将SD卡插入昉·星光 2，启动并执行以下命令查看：

```
ls /home/user
```

Figure 4-6 示例输出



结果：可见文件被正确的拷贝到目标路径下。