StarFive
赛昉科技

昉·星光单板计算机软件技术参考手册

版本：V1.1

日期：2022/02/17

Doc ID：VisionFive-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。 上海赛昉科技有限公司（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： 浦东新区盛夏路61弄张润大厦2号楼502，上海市，201203，中国

网站： http://www.starfivetech.com

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

关于本手册

关于本指南和技术支持信息

简介

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。

修订历史

表 0-1 修订历史

版本	发布说明	修订
V1.0	2021/12/8	首次发布。
V1.1	2022/2/17	更新“移植Rootfs、Kernel和dtb到昉·星光中”的“方法1：使用*Micro SD*卡”章节中步骤12。

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：

- **提示：**
  建议如何在某个主题或步骤中应用信息。
- **注：**
  解释某个特例或阐释一个重要的点。
- **重要：**
  指出与某个主题或步骤有关的重要信息。
- **警告：**
  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。
- **警告：**
  表明某个操作或步骤可能导致物理伤害或硬件损坏。

目录

表格清单

插图清单

1. 硬件准备

请准备如下设备:

- 昉·星光单板计算机单板计算机
- 容量不低于16 GB 的Micro SD卡
- Micro SD卡读卡器
- 计算机（Windows/Mac/Linux）
- USB 转串口转换器（3.3 V I/O）
- 网线
- 电源适配器（5 V / 3 A）
- USB Type-C数据线

注：

本手册中PC主机安装的是Ubuntu 18.04 LTS。

2. 编译固件

如果您对从源代码编译bootloader和ddr init感兴趣，请参考本手册步骤进行编译。

2.1. 准备编译环境

请参考以下步骤准备编译环境：

步骤：

1. 访问此链接下载操作系统相对应的、最新版本的工具链：riscv64-unknown-elf-toolchain-xxx。
2. 执行以下命令解压上一步下载的工具链：

```
tar -xzvf <Toolchain_Name>
```

注：

<Toolchain_Name>是指上一步下载的工具链名称。如`riscv64-unknown-elf-toolchain-10.2.0-2020.12.8-x86_64-linux-ubuntu14.tar.gz`。

3. 执行以下命令打开`.bashrc`文件：

```
gedit ~/.bashrc
```

4. 新增以下代码行，保存后退出：

```
export PATH=$PATH:<Unzipped_Compiler_Path>/bin
```

注：

*<Unzipped_Compiler_Path>*即步骤2中解压出的编译器所在路径，如`/home/yingpeng/Downloads/riscv64-unknown-elf-toolchain-xxx/`。

示例命令：

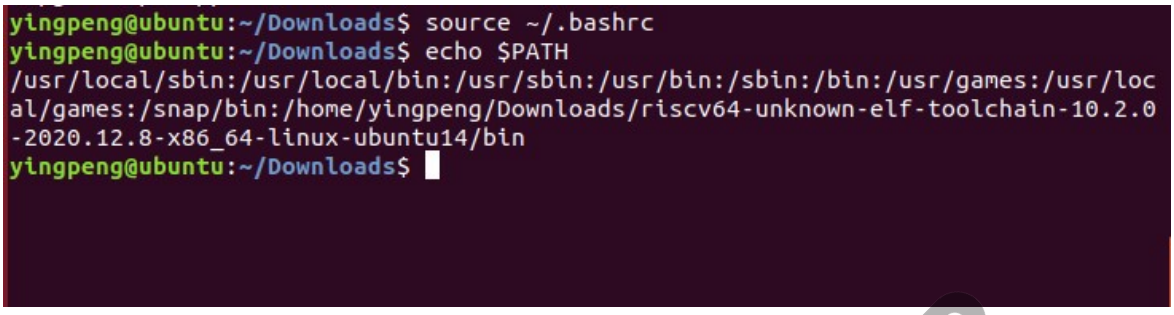
```
export
 PATH=$PATH:/home/yingpeng/Downloads/riscv64-unknown-elf-toolchain-10.2
.0-2020.12.8-x86_64-linux-ubuntu14/bin
```

5. 执行以下命令，使更新立即生效。

```
source ~/.bashrc
```

示例命令及执行结果：

图 2-1 示例



2.2. 编译Bootloader

请参考以下步骤，编译bootloader：

步骤：

1. 执行以下命令，从GitHub复制代码到本地：

```
git clone https://github.com/starfive-tech/JH7100_secondBoot.git
```

2. 进入编译路径：

```
cd JH7100_secondBoot/build
```

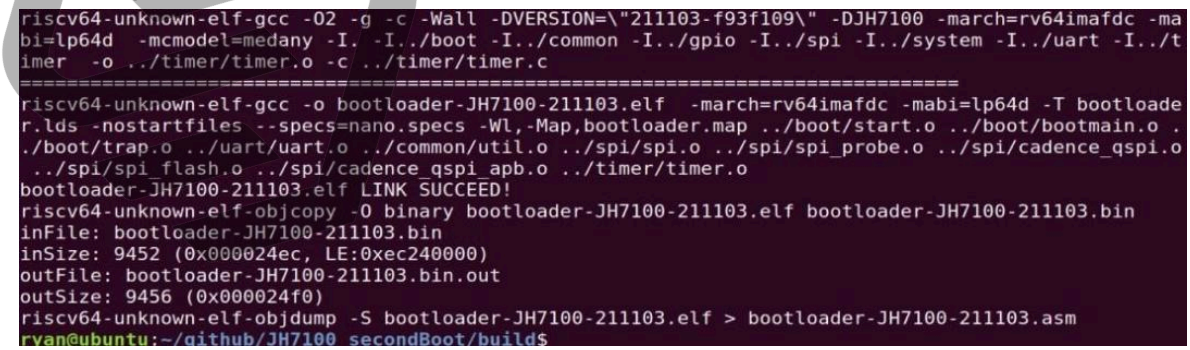
3. 编译bootloader：

```
make
```

结果：

系统输出以下结果，表明您已成功编译bootloader。

图 2-2 示例输出



2.3. 编译ddr init

请参考以下步骤，编译ddr init：

步骤：

1. 执行以下命令，从GitHub复制代码到本地：

```
git clone https://github.com/starfive-tech/JH7100_ddrinit.git
```

2. 进入编译路径：

```
cd JH7100_ddrinit/build
```

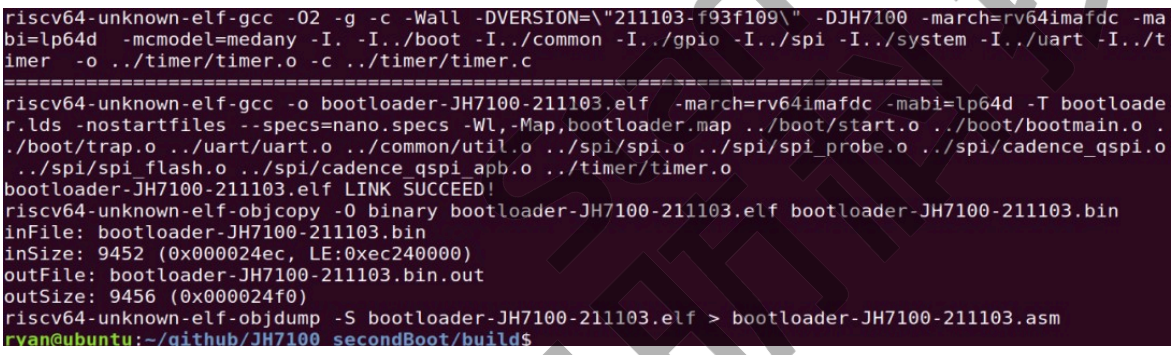
3. 编译ddr init.

```
make
```

结果：

系统输出以下结果，表明您已成功编译ddr init。

图 2-3 示例输出



注：

关于更新bootloader和ddr init的步骤，请参考《昉·星光单板计算机快速参考手册》中“附录 B：更新固件及 U-Boot”章节。

3. 制作通用系统

本章介绍如何制作通用系统。

主要包含以下三个部分：

编译U-boot和Kernel (第 *11*页)

编译Linux Kernel (第 *15*页)

更新Kernel和模块 (第 *16*页)

3.1. 编译U-boot和Kernel

3.1.1. 设置编译环境

请参考以下步骤设置您的交叉编译器：

步骤：

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. 执行以下命令查询riscv64-linux-gnu-gcc编译器版本：

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

结果：

示例输出如下：

图 3-1 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)
```

3.1.2. 编译U-boot

请参考以下步骤编译u-boot:

步骤:

1. 进入到您将用来存放u-boot文件的文件夹，如home directory（主目录）:

   示例命令:

   ```
   cd ~ # home directory
   ```

2. 下载用于编译u-boot的源代码:

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

3. 执行以下命令切换代码分支:

   ```
   cd u-boot
   git checkout -b JH7100_upstream origin/JH7100_upstream
   git pull
   ```

4. 在u-boot目录下执行以下命令编译u-boot:

   ```
   make <Configuration_File> ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
   make u-boot.bin u-boot.dtb ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu-
   ```

   注:

   <Configuration_File> 为配置文件名称，针对不同单板计算机的值为:

   - 昉·星光单板计算机：starfive_jh7100_visionfive_smode_defconfig。
   - 星光板：starfive_jh7100_starlight_smode_defconfig。

结果:

如图所示，编译完成后，u-boot路径下会生成两个文件：u-boot.bin和u-boot.dtb。

图 3-2 示例输出

```
-rwxrwxr-x   1 ryan ryan 7224016 Nov  9 18:14 u-boot
-rw-rw-r--   1 ryan ryan  921147 Nov  9 18:14 u-boot.bin
-rw-rw-r--   1 ryan ryan      47 Nov  9 18:14 .u-boot.bin.cmd
-rw-rw-r--   1 ryan ryan   15585 Nov  9 18:14 u-boot.cfg
-rw-rw-r--   1 ryan ryan     953 Nov  9 18:14 .u-boot.cmd
-rw-rw-r--   1 ryan ryan   41643 Nov  9 18:14 u-boot.dtb
-rw-rw-r--   1 ryan ryan  921147 Nov  9 18:14 u-boot-dtb.bin
```

注:

此后OpenSBI编译将使用u-boot.dtb和u-boot.bin文件。

3.1.3. 编译OpenSBI

OpenSBI全称为Open-source Supervisor Binary Interface，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：

图 3-3 典型的启动流程

请参考以下步骤编译OpenSBI：

步骤：

1. 进入到您将用来存放OpenSBI文件的文件夹，如home directory（主目录）：

   示例：

   ```
   cd ~ # home directory
   ```

2. 下载OpenSBI所需的源代码：

   ```
   git clone https://github.com/riscv/opensbi.git
   ```

3. 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}/u-boot.dtb
   ```

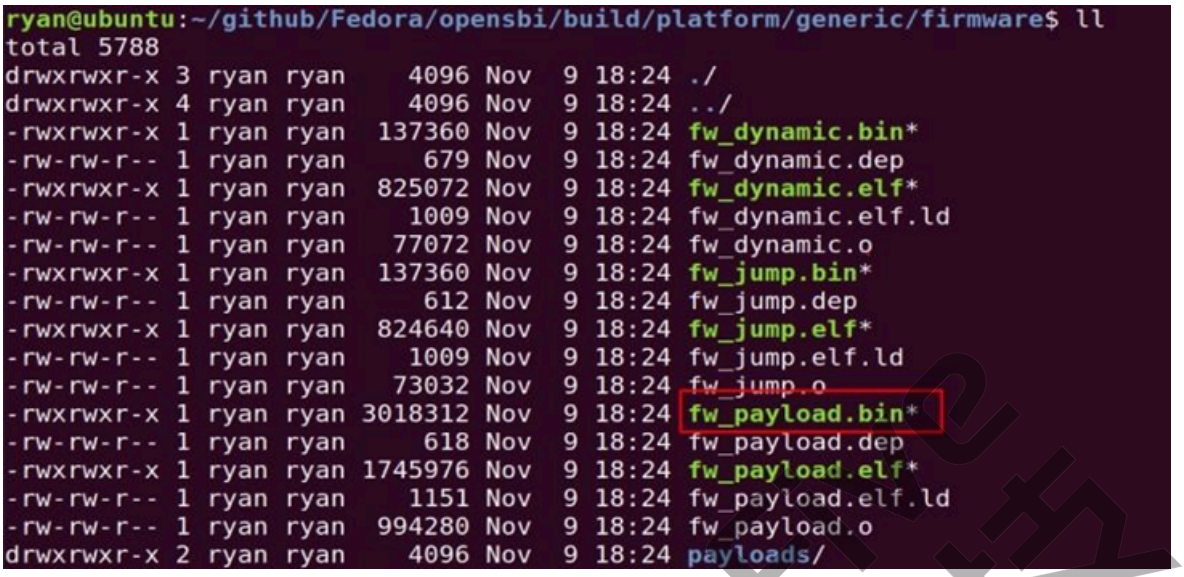
   注：

   将{U-BOOT_PATH}修改为此前存放u-boot文件的路径。

   结果：

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

图 3-4 示例输出



4. 进入到`fw_payload.bin`所在文件夹。

```
cd opensbi/build/platform/generic/firmware
```

5. 复制`fw_payload.bin`到另一文件夹。

```
cp fw_payload.bin ~/Desktop/payload/
```

6. 进入到上一步复制`fw_payload.bin`的文件夹，并执行以下命令，安装镜像转换工具：

```
cd ~ /Desktop/payload/
sudo apt install subversion
svn export
 https://github.com/starfive-tech/freelight-u-sdk.git/branches/starfive
/fsz.sh
```

注：

点击此处可下载源代码。

7. 改变工具的用户权限：

```
chmod 777 fsz.sh
```

8. 将`fw_payload.bin`转换为`fw_payload.bin.out`。

```
./fsz.sh fw_payload.bin fw_payload.bin.out
```

图 3-5 示例输出

```
ryan@ubuntu:~/Desktop/payload$ ./fsz.sh fw_payload.bin fw_payload.bin.out
inFile: fw_payload.bin
inSize: 3018312 (0x002e0e48, LE:0x480e2e00)
outFile: fw_payload.bin.out
outSize: 3018316 (0x002e0e4c)
```

注：

系统将生成fw_payload.bin.out文件。若需要烧录u-boot，请参考《昉·星光单板计算机快速参考手册》中“5 附录 B：更新固件及 U-Boot”章节。

3.2. 编译Linux Kernel

请参考以下步骤，编译Linux Kernel：

1. 进入存放Linux Kernel文件的文件夹，如home directory（主目录）：

   示例命令：

   ```
   cd ~ # home directory
   ```

2. 下载Linux Kernel源代码：

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

3. 输入以下命令设置编译Linux Kernel的默认设置：

   ```
   cd linux
   make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv visionfive_defconfig
   ```

4. 输入以下命令设置编译Linux Kernel的其他设置：

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

5. 编译Linux Kernel：

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

   注：

   按照CPU核的数量，调整此命令中`-jx`的值。如CPU为8核，则将`-jx`改为`-j8`。整个编译过程将耗费一定时间，请耐心等候。

结果：

- 系统将在linux/arch/riscv/boot目录下，生成Kernel镜像文件Image.gz。

图 3-6 示例输出

```
yingpeng@ubuntu:~/Desktop/github/linux/arch/riscv/boot$ ll
total 17628
drwxrwxr-x  3 yingpeng yingpeng     4096 Nov 22 16:18 ./
drwxrwxr-x 11 yingpeng yingpeng     4096 Nov 22 16:15 ../
drwxrwxr-x  6 yingpeng yingpeng     4096 Nov 18 18:52 dts/
-rw-rw-r--  1 yingpeng yingpeng       83 Nov 18 18:52 .gitignore
-rwxrwxr-x  1 yingpeng yingpeng 19723776 Nov 22 16:18 Image*
-rw-rw-r--  1 yingpeng yingpeng      151 Nov 22 16:18 .Image.cmd
-rw-rw-r--  1 yingpeng yingpeng  6353268 Nov 22 16:18 Image.gz
-rw-rw-r--  1 yingpeng yingpeng      101 Nov 22 16:18 .Image.gz.cmd
-rw-rw-r--  1 yingpeng yingpeng     1561 Nov 18 18:52 install.sh
-rw-rw-r--  1 yingpeng yingpeng      206 Nov 18 18:52 loader.lds.S
-rw-rw-r--  1 yingpeng yingpeng      143 Nov 18 18:52 loader.S
-rw-rw-r--  1 yingpeng yingpeng     1612 Nov 18 18:52 Makefile
```

- 系统将在linux/arch/riscv/boot/dts/starfive下，生成dtb文件。

图 3-7 生成dtb文件

```
yingpeng@ubuntu:~/Desktop/github/linux/arch/riscv/boot/dts/starfive$ ls
jh7100-beaglev-starlight-a1.dtb  jh7100.dtsi
jh7100-beaglev-starlight-a1.dts  jh7100-starfive-visionfive-v1.dtb
jh7100-beaglev-starlight.dtb     jh7100-starfive-visionfive-v1.dts
jh7100-beaglev-starlight.dts     Makefile
jh7100-common.dtsi
```

在移植rootfs、dtb和Kernel到昉·星光上时，将使用到Image.gz和.dtb文件。不同的单板计算机将使用不同的dtb文件，详细请参考《赛昉科技40-Pin GPIO Header用户指南》中描述dtb文件的表格。

3.3. 更新Kernel和模块

本章介绍如何更新Kernel和模块。

主要包含以下两个部分：

- 烧录操作系统（Fedora OS）(第 16页)
- 增加新文件 (第 17页)

3.3.1. 烧录操作系统（以Fedora为例）

步骤：

1. 访问赛昉科技官方GitHub下载最新的操作系统。
2. 烧录最新版本的操作系统到Micro SD卡上。详细步骤请参考《昉·星光单板计算机快速参考手册》中的“将 Fedora 烧录到 Micro SD 卡上”章节。

3.3.2. 增加新文件

请参考以下步骤，增加新文件：

注：

若无更新，则无需增加新文件。

1. 在Linux目录下，输入以下命令编译文件：

```
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
INSTALL_PATH=<ROOTFS_PATH> zinstall -<jx>
```

注：

- *<ROOTFS_PATH>*：值为用户自定义的目录，该目录将用于存放生成的vmliunz文件。
- *<jx>*：值为CPU核的数量。如CPU为8核，则值为-j8。

示例命令及输出：

图 3-8 示例命令及输出

```
yingpeng@ubuntu:~/Desktop/github/linux$ make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
INSTALL_PATH=~/Desktop/github/boot zinstall -j4
sh ./arch/riscv/boot/install.sh 5.16.0-rc2-visionfive-g6e924cb10a60 \
arch/riscv/boot/Image.gz System.map "/home/yingpeng/Desktop/github/boot"
yingpeng@ubuntu:~/Desktop/github/linux$
```

结果：

系统将于<ROOTFS_PATH>下，生成vmliunz文件。

2. （可选）在Linux路径下，输入以下指令编译文件：

```
make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
INSTALL_MOD_PATH=<ROOTFS_PATH> modules_install -jx
```

注：

若需增加新模块，则需执行本步骤。

- *<ROOTFS_PATH>*：值为用户自定义的目录，该目录将用于存放生成的模块文件。
- *<jx>*：值为CPU核的数量。如CPU为8核，则值为-j8。

示例命令与输出：

图 3-9 示例命令与输出

```
yingpeng@ubuntu:~/Desktop/github/linux$ make CROSS_COMPILE=riscv64-linux-gnu- ARCH=riscv
INSTALL_MOD_PATH=~/Desktop/github/boot modules_install -j4
  DEPMOD  /home/yingpeng/Desktop/github/boot/lib/modules/5.16.0-rc2-visionfive-g6e924cb10a60
yingpeng@ubuntu:~/Desktop/github/linux$
```

3. 在<*ROOTFS_PATH*>路径下，查看生成的文件。如下图所示：

图 3-10 示例

```
yingpeng@ubuntu:~/Desktop/github/boot$ ls
config-5.16.0-rc2-visionfive-g6e924cb10a60
lib
System.map-5.16.0-rc2-visionfive-g6e924cb10a60
vmlinuz-5.16.0-rc2-visionfive-g6e924cb10a60
yingpeng@ubuntu:~/Desktop/github/boot$
```

4. 增加新文件：

子步骤：

a. 查看Micro SD卡的信息：

```
df -h
```

图 3-11 示例Micro SD卡信息

```
/dev/sdb2       122M  5.6M  117M   5% /media/yingpeng/2AAA-E3BE
/dev/sdb3       458M   86M  358M  20% /media/yingpeng/__boot
/dev/sdb4        12G  7.9G  3.3G  72% /media/yingpeng/__
```

b. 在<*ROOTFS_PATH*>路径下，执行以下指令复制Kernel文件到Micro-SD卡上：

```
sudo cp vmlinuz-5.16.0-rc2-visionfive-g6e924cb10a60
/media/<User_Name>/__boot/ && sync
```

注：

<User_Name> 为您的用户名，如yingpeng。

c. （可选）在<*ROOTFS_PATH*>路径下执行以下命令复制模块文件到Micro SD卡上：

```
sudo cp -r lib/modules/5.16.0-rc2-visionfive-g6e924cb10a60/
/media/<User_Name>/__/lib/modules && sync
```

注：

- 若您已经按照“增加新文件 (第 17页)”章节的步骤2，编译了新模块，则需执行本步骤。
- *<User_Name>*为您的用户名，如yingpeng。

5. 执行以下指令更新grub.cfg文件：

```
cd /media/<UserName>/__boot/
sudo gedit grub.cfg
```

注：

*<User_Name>*为您的用户名，如yingpeng。

6. 新增以下命令行，保存后退出：

```
menuentry '<Configuration_Item_Name_on_Menu>' {
linux /<Newly_Compiled_vmlinuz_file> ro
 root=UUID=59fcd098-2f22-441a-ba45-4f7185baf23f rhgb console=tty0
 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1
 selinux=0 LANG=en_US.UTF-8
devicetree /dtbs/5.16.0-rc2+/starfive/<dtb_File_Name>
initrd /initramfs-5.16.0-rc2+.img
}
```

注：

以上命令中：

- *<Configuration_Item_Name_on_Menu>*：菜单上显示的配置项名称，如My Fedora vmlinux-5.16.0-rc2 visionfive。
- *<Newly_Compiled_vmlinuz_file>*：编译后的vmlinuz文件名称，如vmlinuz-5.16.0-rc2-visionfive-g6e924cb10a60。
- *<dtb_File_Name>*：dtb文件名。不同的单板计算机将使用到不同的dtb文件，详细请参考《赛昉科技40-Pin GPIO Header用户指南》中描述dtb文件的表格。

示例：

以下为示例命令行：

```
menuentry 'My Fedora vmlinux-5.16.0-rc2 visionfive' {
linux /vmlinuz-5.16.0-rc2-visionfive-g6e924cb10a60 ro
 root=UUID=59fcd098-2f22-441a-ba45-4f7185baf23f rhgb console=tty0
```

```
 console=ttyS0,115200 earlycon rootwait stmmaceth=chain_mode:1 selinux=0
 LANG=en_US.UTF-8
devicetree /dtbs/5.16.0-rc2+/starfive/jh7100-starfive-visionfive-v1.dtb
initrd /initramfs-5.16.0-rc2+.img
}
```

验证：

请参考以下步骤，验证设置是否成功：

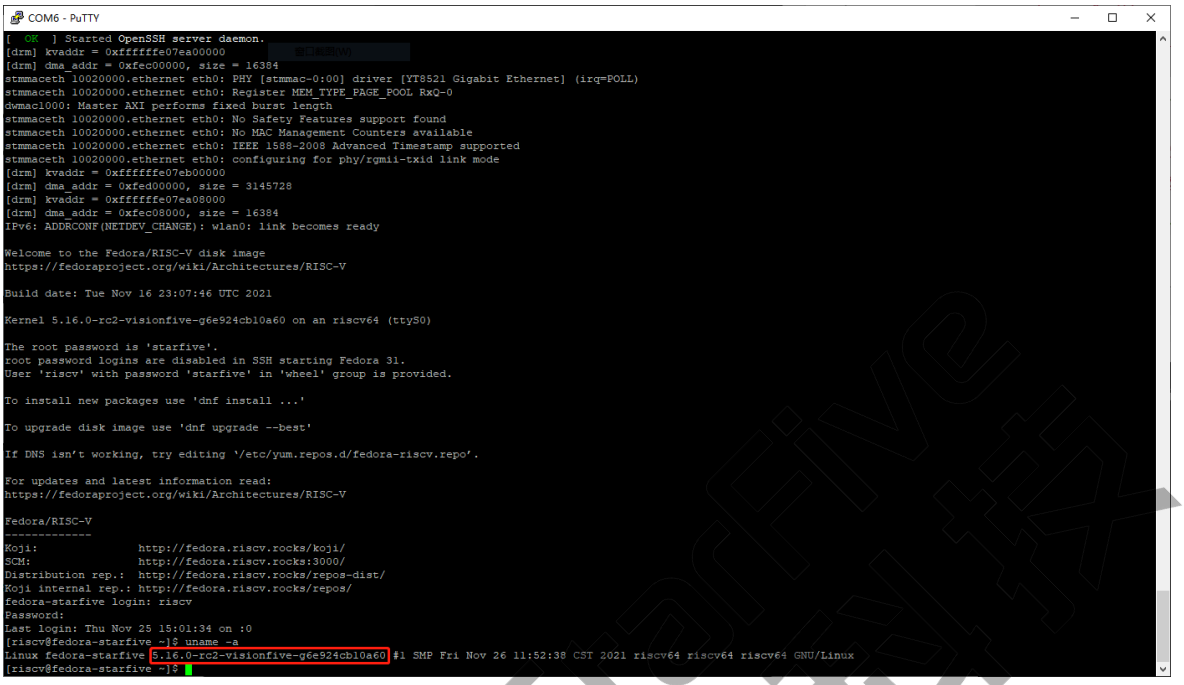
1. 从电脑中拔出Micro SD卡，并插入到昉·星光单板计算机。上电后系统将正常启动。
2. 菜单将显示此前定义的设置项，如**My Fedora vmlinux-5.16.0-rc2 visionfive**。

   如下图所示：

图 3-12 示例界面

3. 系统成功启动后，将显示编译后的vmlinuz文件版本：

图 3-13 示例输出



4. 制作Busybox系统

本章介绍如何制作busybox系统。

主要包含以下两部分：

4.1. 制作文件系统

请参考以下步骤，制作文件系统：

步骤：

1. 创建目录结构：

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

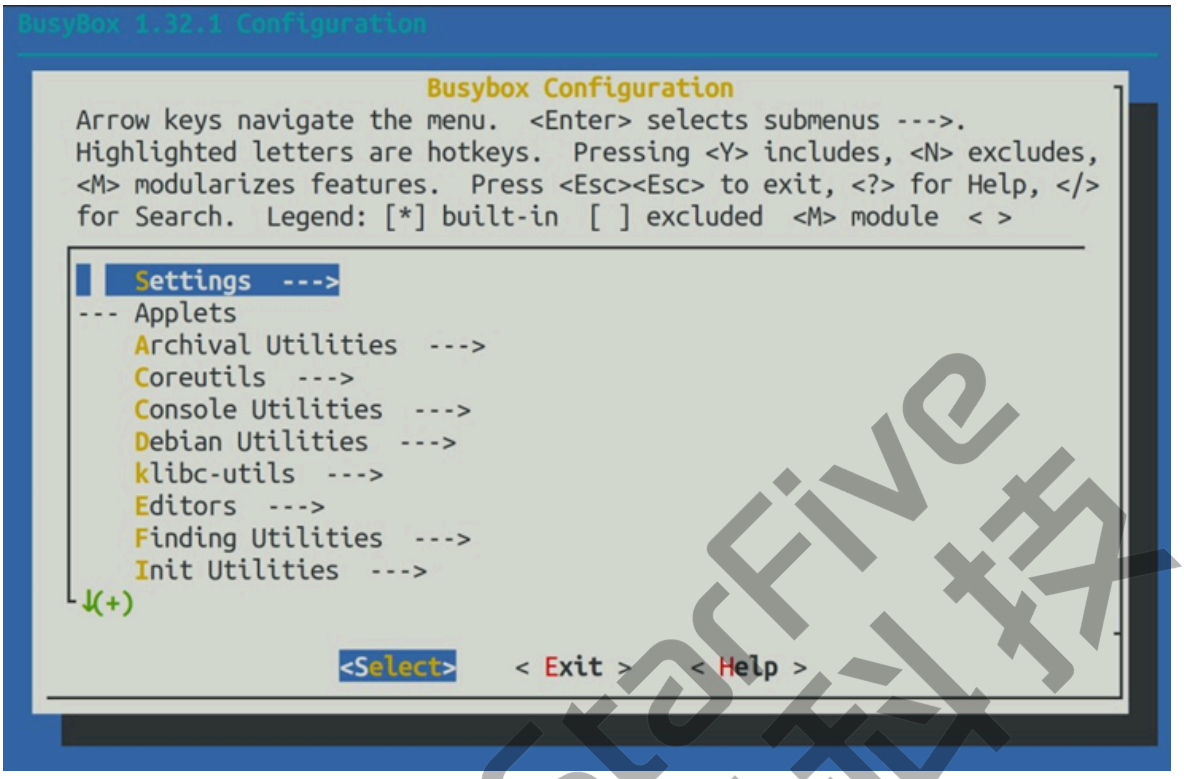
2. 下载busybox源代码，保存至rootfs文件夹以外的路径：

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

3. 进入解压文件所在文件夹，并进入busybox设置界面：

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

图 4-1 Busybox设置界面



4. 选择**Settings** > **Build Options**，按**Y**键查看Build static binary (no shared libs)选项。
5. 在**Build Options**下选择**cross compiler prefix**，并输入以下命令指定编译器：

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

6. 选择**Installation Options** > **Destination path for 'make install'**，根据提示输入以下路径，将其设置为rootfs文件路径（即编译后的busybox安装路径）。

   示例：

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

7. 退出busybox设置窗口，并保存设置。
8. 编译busybox。

```
make ARCH=riscv
```

9. 安装busybox：

```
make install
```

10. 进入到此前创建的`rootfs/etc`目录，新建名为inittab的文件，然后使用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`路径下，新建名为profile的文件，然后使用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`路径下，新建名为fstab的文件，然后使用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`路径下新建名为passwd的文件，然后使用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`路径下新建名为group的文件，然后使用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`路径下新建名为shadow的文件，然后使用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`路径下新建名为init.d的目录，然后进入到该目录：

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

23. 在`rootfs/etc/init.d`路径下，新建名为rcS的文件，然后使用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. 进入到此前创建的rootfs/dev路径下，并执行以下命令：

```
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根目录下，新建软链接：

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

27. 修改rootfs目录下所有文件的权限：

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

28. 在rootfs目录下，执行以下命令在指定的目录下生成rootfs.cpio.gz（cpio系统软件包）。

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

注：

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。整个编译过程需要耗费一定时间，请耐心等待。

4.2. 移植Rootfs、Kernel和dtb到昉·星光单板计算机

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

图 4-2 示例界面

注：

请注意不同的单板计算机将使用到不同的dtb文件，详细请参考《赛昉科技40-Pin GPIO Header用户指南》中描述dtb文件的表格。

4.2.1. 方法1：使用Micro SD卡

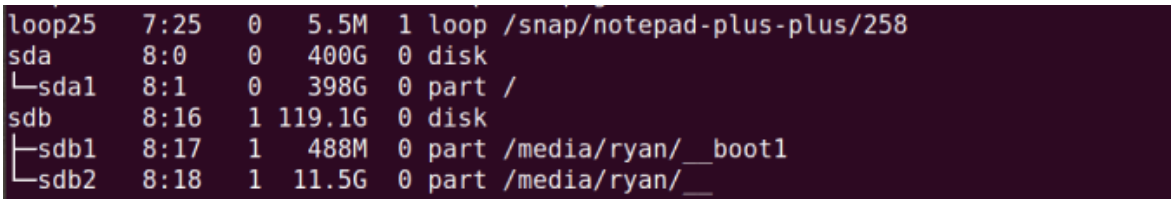
步骤：

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`。

图 4-3 示例



3. 输入以下命令，进入分区配置：

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

图 4-4 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
```

提示：
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：

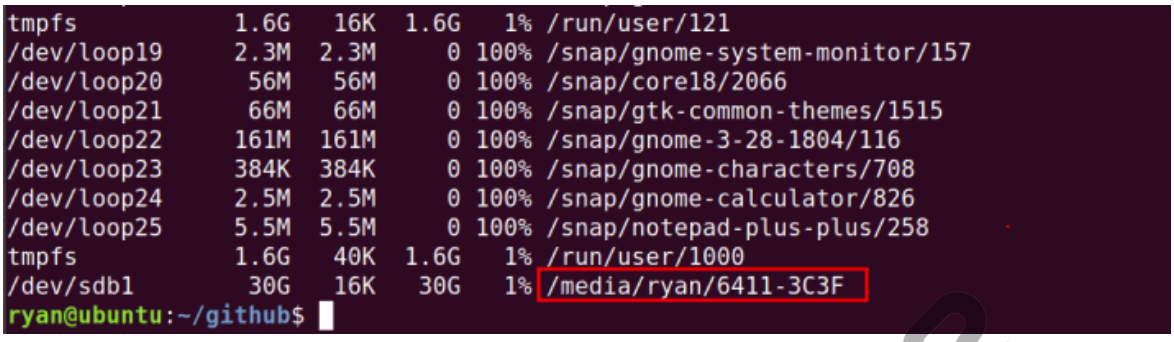
```
sudo mkfs.vfat /dev/sdb1
```

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。

图 4-5 示例输出



8. 进入到 rootfs 系统软件包、Kernel 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
```

注：

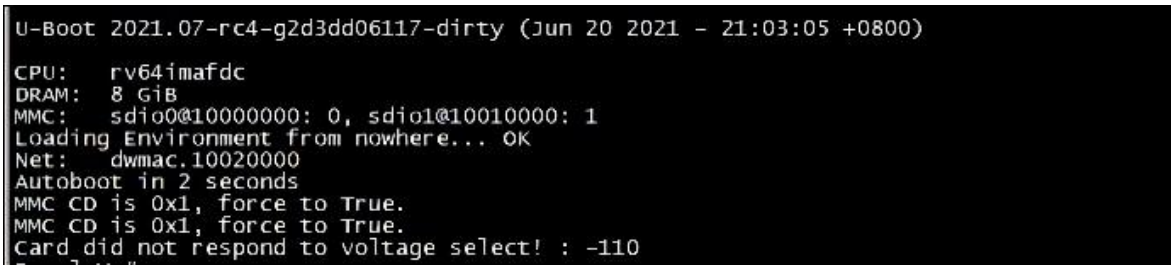
- *<Mount_Location>*指此前记录的挂载路径。
- *<dtb_File_Name>*指dtb文件名。请注意不同的单板计算机将使用到不同的dtb文件，详细请参考《赛昉科技40-Pin GPIO Header用户指南》中描述dtb文件的表格。

示例命令:

```
sudo cp Image.gz /media/user/6411-3C3F/
sudo cp rootfs.cpio.gz /media/user/6411-3C3F/
sudo cp jh7100-starfive-visionfive-v1.dtb /media/user/6411-3C3F/
sync
```

10. 从计算机中移除Micro SD 卡，并将该卡插入昉·星光单板计算机，然后启动昉·星光。
11. 使用 USB 转串口转换器，将昉·星光连接至计算机，然后打开minicom，等待昉·星光单板计算机进入 **u-boot** 模式。以下示例输出表明昉·星光已进入u-boot 模式：

图 4-6 示例输出



12. 输入以下命令：

```
setenv kernel_comp_addr_r 0x90000000;setenv kernel_comp_size
 0x10000000;setenv kernel_addr_r 0x84000000;setenv fdt_addr_r
 0x88000000;setenv ramdisk_addr_r 0x88300000
fatls mmc 0:1
fatload mmc 0:1 ${kernel_addr_r} Image.gz
fatload mmc 0:1 ${fdt_addr_r} jh7100-starfive-visionfive-v1.dtb
fatload mmc 0:1 ${ramdisk_addr_r} rootfs.cpio.gz
booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r}
```

13. 输入以下用户名和密码登录：

    - **Username:** root
    - **Password:** starfive

4.2.2. 方法2：使用网线

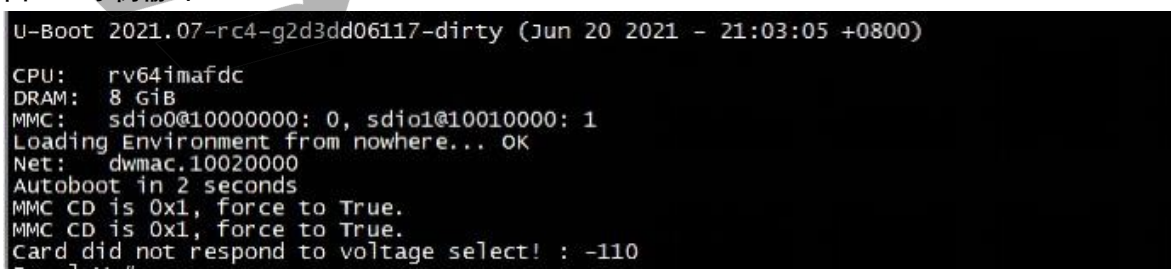
1. 使用网线通过昉·星光上的RJ45接口和路由器连接，连接串口转换器，然后启动昉·星光。

   注：

   请确保计算机也通过网线或WiFi与路由器连接。

2. 打开minicom等待昉·星光进入u-boot模式。以下示例输出表明昉·星光已进入u-boot模式：

图 4-7 示例输出



3. 输入以下指令设置u-boot环境变量：

```
setenv serverip 192.168.120.12;setenv ipaddr 192.168.120.200;
setenv hostname starfive;setenv netdev eth0;
setenv kernel_comp_addr_r 0x90000000;setenv kernel_comp_size
 0x10000000;
setenv bootargs console=ttyS0,115200 earlycon=sbi root=/dev/ram0
 stmmaceth=chain_mode:1 loglevel=8
```

注：

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·星光的 IP 地址应为 192.168.120.xxx。但是， 如果您的路由器IP不同（例如 192.168.2.1），请确保服务器 IP 和昉·星光属于同一 IP 段（例如 192.168.2.xxx）中。

4. 通过ping 命令来检查主机与昉·星光的连接情况：

示例命令：

```
ping 192.168.120.12
```

结果：

以下输出表明主机与昉·星光已经在同一网络下建立连接。

图 4-8 示例输出

```
VisionFive #ping 192.168.120.12
Speed: 1000, full duplex
Using dwmac.10020000 device
host 192.168.120.12 is alive
VisionFive #
```

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. 输入以下指令，进入TFTP服务器设置：

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

8. 输入以下指令设置TFTP服务器：

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/home/user/Desktop/compiled"
```

```
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

注：

命令中TFTP_DIRECTORY是三个镜像文件（Image.gz，jh7100-starfive-visionfive-v1.dtb, rootfs.cpio.gz）所在的路径。

9. 重启TFTP服务器。

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

10. 在昉·星光的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Kernel：

```
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}
```

注：

*<dtb_File_Name>*指dtb文件名。请注意不同的单板计算机将使用到不同的dtb文件，详细请参考《赛昉科技40-Pin GPIO Header用户指南》中描述dtb文件的表格。

示例命令：

```
tftpboot ${fdt_addr_r} jh7100-starfive-visionfive-v1.dtb;tftpboot
 ${kernel_addr_r} Image.gz;tftpboot ${ramdisk_addr_r}
 rootfs.cpio.gz;booti ${kernel_addr_r} ${ramdisk_addr_r}:${filesize}
 ${fdt_addr_r}
```

11. 使用以下用户名密码登录：

Username: root

Password: starfive