# 简单字符驱动模块挂载

字符驱动是linux常见、基本的一类设备驱动，一般是按字节顺序进行读写操作。

例如LED，按键，IIC，SPI等设备。

linux的ko驱动有两种方式运行，一是编译进内核，开机自启动，二是编译ko模块，需要用时加载模块运行。

本文介绍的是编译出.ko模块去加载模块运行。个人认为编译ko模块会比编译进内核灵活方便使用。也可以写一个开机脚本去载入内核。所以我们使用的时候更倾向于ko模块，这个个人理解，如有不同看法可以评论交流。

*注意以下的所有操作都需要使用root权限！！！*

*权限提升 sudo -i*

## 安装必要的包

# Debian/Ubuntu

sudo apt-get install libncurses-dev libssl-dev bc flex bison make gcc gcc-riscv64-linux-gnu

# Fedora

sudo dnf install ncurses-devel openssl openssl-devel bc flex bison make gcc gcc-riscv64-linux-gnu

# Archlinux

sudo pacman -S --needed ncurses openssl bc flex bison make gcc riscv64-linux-gnu-gcc

按照自己的系统进行安装包。

## 获取内核

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

### 2.1切换分支

cd linux

git checkout -b JH7110\_VisionFive2\_devel origin/JH7110\_VisionFive2\_devel

git pull

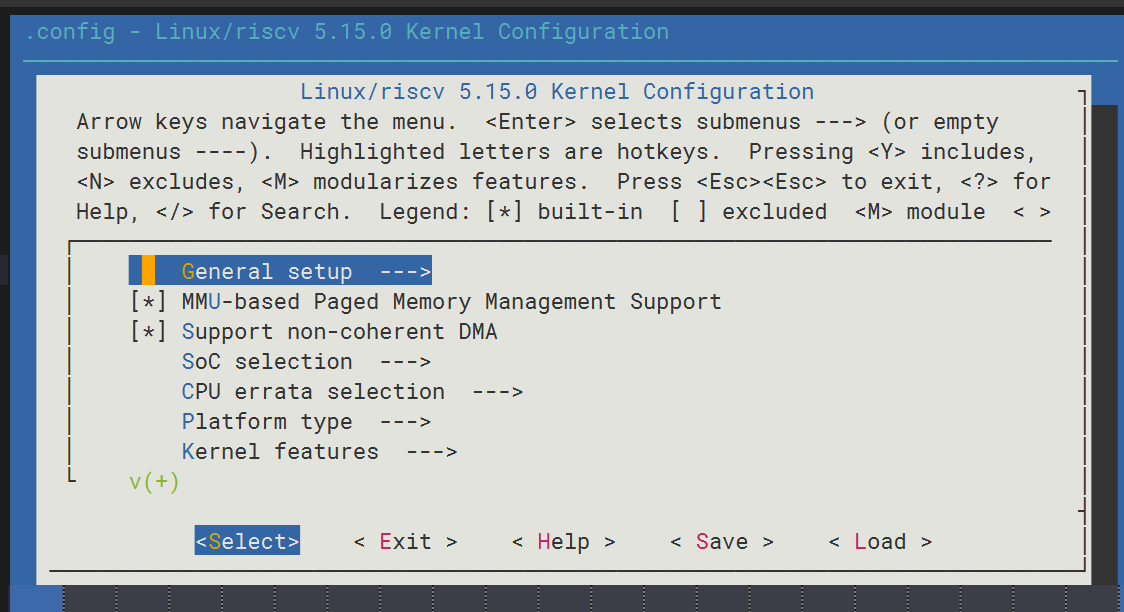
## 设置编译内核

make starfive\_visionfive2\_defconfig CROSS\_COMPILE=riscv64-linux-gnu- ARCH=riscv

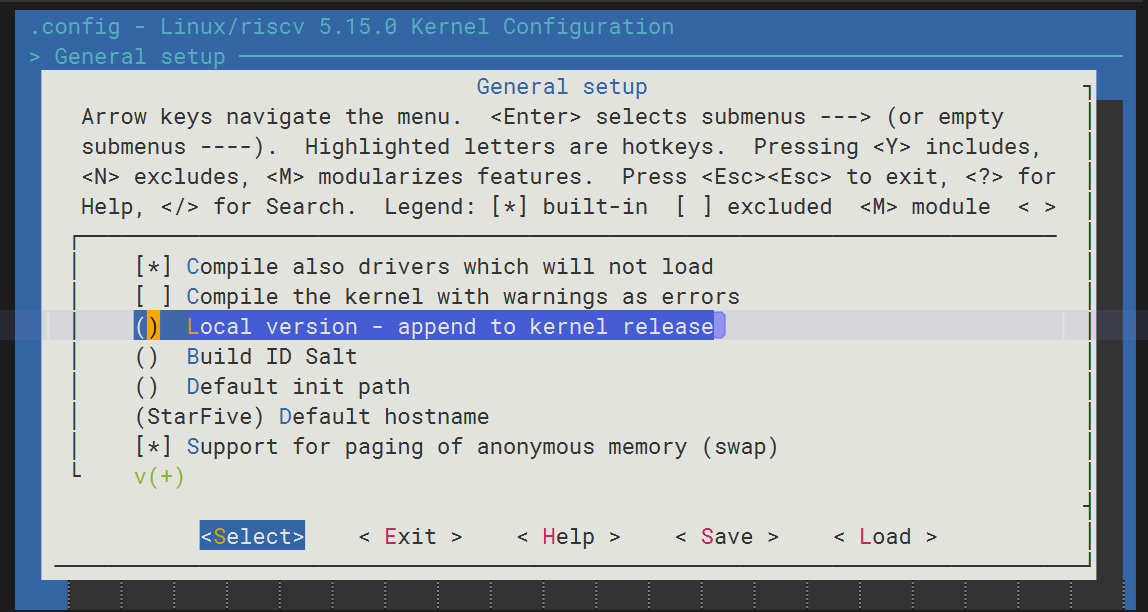
## 设置linux内核

make CROSS\_COMPILE=riscv64-linux-gnu- ARCH=riscv menuconfig -j2

打开menuconfig设置



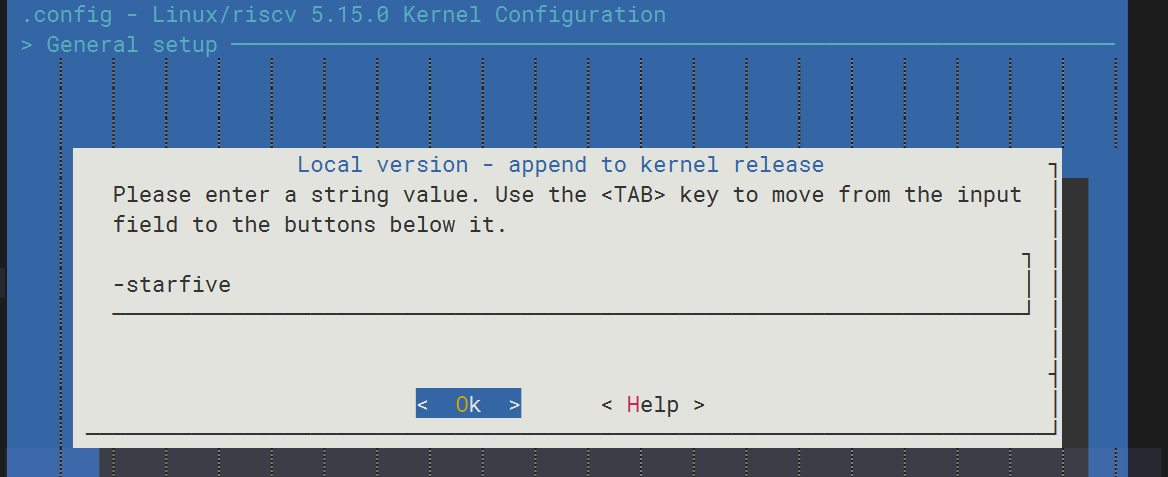
选择第一个general setup



选择第三个

Local version - append to kernel release

输入 -starfive



保存设置，并退出。

这个操作的目的是为了修改这个内核的名字，防止加载模块时报 内核不匹配的错误。

## 5.编译内核。

make CROSS\_COMPILE=riscv64-linux-gnu- ARCH=riscv -j2

## 6.在创建一个文件夹，里面保存两个文件

进入该文件夹

mkdir char\_dev\_driver

cd ./char\_dev\_driver

### 6.1第一个char\_dev\_driver.c ：

#include <linux/init.h>

#include <linux/kernel.h>

#include <linux/module.h>

/\* 驱动入口函数 \*/

static int \_\_init char\_dev\_driver\_init(void)

{

/\* 入口函数具体内容 \*/

printk("char\_dev\_driver\_init...\n");

return 0;

}

/\* 驱动出口函数 \*/

static void \_\_exit char\_dev\_driver\_exit(void)

{

/\* 出口函数具体内容 \*/

printk("char\_dev\_driver\_exit...\n");

}

module\_init(char\_dev\_driver\_init); // 注册模块加载函数

module\_exit(char\_dev\_driver\_exit); // 注册模块卸载函数

MODULE\_LICENSE("GPL"); // license

MODULE\_AUTHOR("Wang Rongwen"); // author

MODULE\_DESCRIPTION("A simple Hello World Module");

MODULE\_ALIAS("a simplest module");

### 6.2第二个Makefile ：

KERNEL\_DIR :=/home/w/linux/ #linux内核路径

CURRENT\_DIR := $(shell pwd)

obj-m := char\_dev\_driver.o

build: kernel\_modules

kernel\_modules:

$(MAKE) -C $(KERNEL\_DIR) M=$(CURRENT\_DIR) modules

clean:

$(MAKE) -C $(KERNEL\_DIR) M=$(CURRENT\_DIR) clean

注意KERNEL\_DIR变量，要填刚刚编译的linux内核路径。

## 7.编译ko模块文件

拷贝到当前的/lib/modules/#uname -r#/下。

编译ko模块，

make -j ARCH=riscv CROSS\_COMPILE=riscv64-linux-gnu-

复制到/lib/modules/#uname -r#/下，以VF2为例：

cp ./char\_dev\_driver.ko /lib/modules/5.15.0-starfive/

清理编译

make clean

## 8.进入该路径并载入模块

## 8.1 复制.ko 文件

编译得到的.ko 文件需要复制到  
/lib/modules/#uname -r#的目录下。

cd /lib/modules/5.15.0-starfive/

### 8.2 depmod

加载模块的依赖

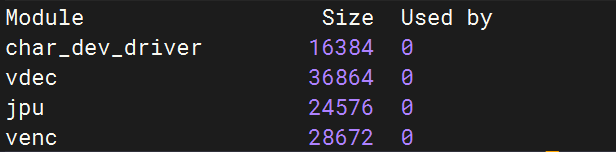
### 8.3 modprobe xxx

加载xxx模块到内核中

modprobe char\_dev\_driver

### 8.4 lsmod

查看linux所有加载进内核的模块

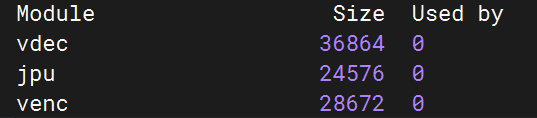


### 8.5 rmmod xxx

卸载内核

rmmod char\_dev\_driver

使用lsmod查看模块



## 9查看操作

使用dmesg查看是否完成操作。



参考地址：

<https://github.com/starfive-tech/linux>

https://doc.rvspace.org/VisionFive2/SW\_TRM/VisionFive2\_SW\_TRM/swtrm\_compiling\_linux\_kernel%20-%20vf2.html

# 在SD卡和eMMC上根目录扩容

在使用apt install指令时，发现安装不了，使用 df -h检测系统内存，发现根目录使用达到了100%，故需要对根目录内存进行扩容。

官方扩容的教程地址：<https://doc.rvspace.org/VisionFive2/Quick_Start_Guide/VisionFive2_QSG/extend_partition.html>

 总结：这是VF2的模块挂载操作，内核最大的问题就是官方下载的镜像和github上的linux内核名字不一致，需要在编译的时候提供相同的名字，否则就会挂载失败。总体来说在内核层面上的操作与ARM核的操作差不多，对应模块这一方面还是非常完善的