龙芯中科
LOONGSON TECHNOLOGY

广东龙芯 2K0300 先锋派用户手册

广东龙芯中科电子科技有限公司
广州市黄埔区神舟路 18 号润慧科技园 B 栋 4 楼龙芯中科 020-22900688

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本已更新的内容。

修订日期	版本	修订者	修订说明
2024/8/1	V0.9	广东龙芯技术部	文档创建
2024/8/8	V0.91	广东龙芯技术部	修改部分描述错误
2024/8/16	V0.92	广东龙芯技术部	修改部分描述错误，更新功能测试的图片
2024/9/4	V0.93	广东龙芯技术部	更新部分描述

目录

一、LoongArch 介绍

1.0 LoongArch简介

LoongArch(中文名龙架构)是龙芯中科于2021年推出的一种新的RISC ISA。LoongArch指令集包括一个 32位版（LA32）、一个64位版（LA64）。LoongArch定义了四个特权级（PLV0~PLV3），其中PLV0是 最高特权级，用于内核；而PLV3 是最低特权级，用于应用程序。龙架构采用基础部分加扩展部分的模块 化组织形式。一个兼容龙架构的 CPU，除实现必需的基础部分（Loongson Base， LBase）外，可根据 实际需求选择实现各扩展部分。目前龙架构已定义的扩展部分包括：虚拟化扩展（Loongson Virtualization， LVZ）、二进制翻译扩展（Loongson Binary Translation， LBT）、128 位向量扩展 （Loongson SIMD Extension， LSX）和 256 位高级向量扩展（Loongson Advanced SIMD Extension， LASX）。本文档介绍了LoongArch的寄存器、基础指令集、虚拟内存以及其他一些主题。

1.1 寄存器

LoongArch的寄存器包括通用寄存器（GPRs）、浮点寄存器（FPRs）、向量寄存器（VRs） 和用于特权 模式（PLV0）的控制状态寄存器（CSRs）。

1.1.1 通用寄存器

LoongArch包括32个通用寄存器（ `$r0` ~ `$r31` ），LA32中每个寄存器为32位宽， LA64中每个寄存器 为64位宽。 `$r0` 的内容总是固定为0，而其他寄存器在体系结构层面 没有特殊功能。（ `$r1` 算是一个 例外，在BL指令中固定用作链接返回寄存器。）

内核使用了一套LoongArch寄存器约定，定义在LoongArch ELF psABI规范中，详细描述参见参考文献:

寄存器名	别名	用途	跨调用保持
`$r0`	`$zero`	常量0	不使用
`$r1`	`$ra`	返回地址	否
`$r2`	`$tp`	TLS/线程信息指针	不使用
`$r3`	`$sp`	栈指针	是
`$r4` `$r11`	`$a0` `$a7`	参数寄存器	否
`$r4` `$r5`	`$v0` `$v1`	返回值	否
`$r12` `$r20`	`$t0` `$t8`	临时寄存器	否
`$r21`	`$u0`	每CPU变量基地址	不使用
`$r22`	`$fp`	帧指针	是
`$r23` `$r31`	`$s0` `$s8`	静态寄存器	是

Note

注意： `$r21` 寄存器在ELF psABI中保留未使用，但是在Linux内核用于保 存每CPU变量基地址。该寄存
器没有ABI命名，不过在内核中称为 `$u0` 。在 一些遗留代码中有时可能见到 `$v0` 和 `$v1` ，它们 是 `$a0` 和 `$a1` 的别名，属于已经废弃的用法。

1.1.2 浮点寄存器

当系统中存在FPU时， LoongArch有32个浮点寄存器（ `$f0` ~ `$f31` ）。在LA64 的CPU核上，每个寄 存器均为64位宽。

浮点寄存器的使用约定与LoongArch ELF psABI规范的描述相同：

寄存器名	别名	用途	跨调用保持
`$f0` `$f7`	`$fa0` `$fa7`	参数寄存器	否
`$f0` `$f1`	`$fv0` `$fv1`	返回值	否
`$f8` `$f23`	`$ft0` `$ft15`	临时寄存器	否
`$f24` `$f31`	`$fs0` `$fs7`	静态寄存器	是

Note

注意：在一些遗留代码中有时可能见到 `$fv0` 和 `$fv1` ，它们是 `$fa0` 和 `$fa1` 的别名，属于已经废弃的用法。

1.1.3 向量寄存器

LoongArch现有两种向量扩展：

- 128位向量扩展LSX（全称Loongson SIMD eXtention），
- 256位向量扩展LASX（全称Loongson Advanced SIMD

eXtention）。 LSX使用 `$v0` ~ `$v31` 向量寄存器，而LASX则使用

`$x0` ~ `$x31` 。

浮点寄存器和向量寄存器是复用的，比如：在一个实现了LSX和LASX的核上， `$x0` 的 低128位与 `$v0` 共用， `$v0` 的低64位与 `$f0` 共用，其他寄存器依此类推。

1.1.4 控制状态寄存器

控制状态寄存器只能在特权模式（PLV0）下访问:

地址	全称描述	简称
0x0	当前模式信息	CRMD
0x1	异常前模式信息	PRMD
0x2	扩展部件使能	EUEN
0x3	杂项控制	MISC
0x4	异常配置	ECFG
0x5	异常状态	ESTAT

地址	全称描述	简称
0x6	异常返回地址	ERA
0x7	出错(Faulting)虚拟地址	BADV
0x8	出错(Faulting)指令字	BADI
0xC	异常入口地址	EENTRY
0x10	TLB索引	TLBIDX
0x11	TLB表项高位	TLBEHI
0x12	TLB表项低位0	TLBELO0
0x13	TLB表项低位1	TLBELO1
0x18	地址空间标识符	ASID
0x19	低半地址空间页全局目录基址	PGDL
0x1A	高半地址空间页全局目录基址	PGDH
0x1B	页全局目录基址	PGD
0x1C	页表遍历控制低半部分	PWCL
0x1D	页表遍历控制高半部分	PWCH
0x1E	STLB页大小	STLBPS
0x1F	缩减虚地址配置	RVACFG
0x20	CPU编号	CPUID
0x21	特权资源配置信息1	PRCFG1
0x22	特权资源配置信息2	PRCFG2
0x23	特权资源配置信息3	PRCFG3
0x30+n (0≤n≤15)	数据保存寄存器	SAVEn
0x40	定时器编号	TID
0x41	定时器配置	TCFG
0x42	定时器值	TVAL
0x43	计时器补偿	CNTC
0x44	定时器中断清除	TICLR
0x60	LLBit相关控制	LLBCTL
0x80	实现相关控制1	IMPCTL1
0x81	实现相关控制2	IMPCTL2

地址	全称描述	简称
0x88	TLB重填异常入口地址	TLBRENTRY
0x89	TLB重填异常出错(Faulting)虚地址	TLBRBADV
0x8A	TLB重填异常返回地址	TLBRERA
0x8B	TLB重填异常数据保存	TLBRSAVE
0x8C	TLB重填异常表项低位0	TLBRELO0
0x8D	TLB重填异常表项低位1	TLBRELO1
0x8E	TLB重填异常表项高位	TLBEHI
0x8F	TLB重填异常前模式信息	TLBRPRMD
0x90	机器错误控制	MERRCTL
0x91	机器错误信息1	MERRINFO1
0x92	机器错误信息2	MERRINFO2
0x93	机器错误异常入口地址	MERRENTRY
0x94	机器错误异常返回地址	MERRERA
0x95	机器错误异常数据保存	MERRSAVE
0x98	高速缓存标签	CTAG
0x180+n (0≤n≤3)	直接映射配置窗口n	DMWn
0x200+2n (0≤n≤31)	性能监测配置n	PMCFGn
0x201+2n (0≤n≤31)	性能监测计数器n	PMCNTn
0x300	内存读写监视点整体控制	MWPC
0x301	内存读写监视点整体状态	MWPS
0x310+8n (0≤n≤7)	内存读写监视点n配置1	MWPnCFG1
0x311+8n (0≤n≤7)	内存读写监视点n配置2	MWPnCFG2
0x312+8n (0≤n≤7)	内存读写监视点n配置3	MWPnCFG3
0x313+8n (0≤n≤7)	内存读写监视点n配置4	MWPnCFG4
0x380	取指监视点整体控制	FWPC
0x381	取指监视点整体状态	FWPS
0x390+8n (0≤n≤7)	取指监视点n配置1	FWPnCFG1
0x391+8n (0≤n≤7)	取指监视点n配置2	FWPnCFG2
0x392+8n (0≤n≤7)	取指监视点n配置3	FWPnCFG3

地址	全称描述	简称
0x393+8n (0≤n≤7)	取指监视点n配置4	FWPnCFG4
0x500	调试寄存器	DBG
0x501	调试异常返回地址	DERA
0x502	调试数据保存	DSAVE

ERA，TLBRERA，MERRERA和DERA有时也分别称为EPC，TLBREPC，MERREPC和DEPC。

1.2 基础指令集

1.2.1 指令格式

LoongArch的指令字长为32位，一共有9种基本指令格式（以及一些变体）：

格式名称	指令构成
2R	Opcode + Rj + Rd
3R	Opcode + Rk + Rj + Rd
4R	Opcode + Ra + Rk + Rj + Rd
2RI8	Opcode + I8 + Rj + Rd
2RI12	Opcode + I12 + Rj + Rd
2RI14	Opcode + I14 + Rj + Rd
2RI16	Opcode + I16 + Rj + Rd
1RI21	Opcode + I21L + Rj + I21H
I26	Opcode + I26L + I26H

Opcode是指令操作码， Rj和Rk是源操作数（寄存器）， Rd是目标操作数（寄存器）， Ra是 4R-type格 式特有的附加操作数（寄存器）。 I8/I12/I14/I16/I21/I26分别是8位/12位/14位/16位/ 21位/26位的立即 数。其中较长的21位和26位立即数在指令字中被分割为高位部分与低位 部分，所以你们在这里的格式描
述中能够看到I21L/I21H和I26L/I26H这样带后缀的表述。

1.2.2 指令列表

为了简便起见，我们在此只罗列一下指令名称（助记符），需要详细信息请阅读 参考文献 中的文档。

1. 算术运算指令:

```
ADD.W SUB.W ADDI.W ADD.D SUB.D

ADDI.D SLT SLTU SLTI SLTUI

AND OR NOR XOR ANDN ORN ANDI ORI XORI
MUL.W  MULH.W  MULH.WU  DIV.W  DIV.WU  MOD.W

MOD.WU  MUL.D  MULH.D  MULH.DU  DIV.D  DIV.DU

MOD.D MOD.DU PCADDI PCADDU12I PCADDU18I

LU12I.W LU32I.D LU52I.D ADDU16I.D
```

2. 移位运算指令:

```
SLL.W SRL.W SRA.W ROTR.W SLLI.W SRLI.W SRAI.W

ROTRI.W SLL.D SRL.D SRA.D ROTR.D SLLI.D SRLI.D SRAI.D

ROTRI.D
```

3. 位域操作指令:

```
EXT.W.B EXT.W.H CLO.W CLO.D SLZ.W CLZ.D CTO.W CTO.D CTZ.W

CTZ.D BYTEPICK.W BYTEPICK.D BSTRINS.W BSTRINS.D BSTRPICK.W

BSTRPICK.D

REVB.2H REVB.4H REVB.2W REVB.D REVH.2W REVH.D BITREV.4B BITREV.8B

BITREV.W BITREV.D

MASKEQZ MASKNEZ
```

4. 分支转移指令:

```
BEQ BNE BLT BGE BLTU BGEU BEQZ BNEZ B BL JIRL
```

5. 访存读写指令:

```
LD.B LD.BU LD.H LD.HU LD.W LD.WU LD.D ST.B ST.H ST.W ST.D
LDX.B LDX.BU LDX.H LDX.HU LDX.W LDX.WU LDX.D STX.B STX.H STX.W STX.D
LDPTR.W LDPTR.D STPTR.W STPTR.D
PRELD PRELDX
```

6. 原子操作指令:

```
LL.W SC.W LL.D SC.D

AMSWAP.W AMSWAP.D AMADD.W AMADD.D AMAND.W AMAND.D AMOR.W

AMOR.D AMXOR.W AMXOR.D
AMMAX.W AMMAX.D AMMIN.W AMMIN.D
```

7. 栅障指令:

```
IBAR DBAR
```

8. 特殊指令:

```
SYSCALL BREAK CPUCFG NOP IDLE ERTN(ERET) DBCL(DBGCALL) RDTIMEL.W
RDTIMEH.W RDTIME.D
ASRTLE.D ASRTGT.D
```

9. 特权指令:

```
CSRRD CSRWR CSRXCHG

IOCSRRD.B IOCSRRD.H IOCSRRD.W IOCSRRD.D IOCSRWR.B IOCSRWR.H

IOCSRWR.W IOCSRWR.D

CACOP TLBP(TLBSRCH) TLBRD TLBWR TLBFILL TLBCLR TLBFLUSH INVTLB LDDIR
LDPTE
```

1.3 虚拟内存

LoongArch可以使用直接映射虚拟内存和分页映射虚拟内存。

直接映射虚拟内存通过CSR.DMWn（ n=0~3）来进行配置，虚拟地址（VA）和物理地址（PA ） 之间有 简单的映射关系:

```
VA = PA + 固定偏移
```

分页映射的虚拟地址（VA）和物理地址（PA）有任意的映射关系，这种关系记录在TLB和页 表中。LoongArch的 TLB包 括 一 个 全 相 联 的 MTLB（ Multiple Page Size TLB， 多 样 页 大 小 TLB ） 和 一 个 组 相 联 的STLB（Single Page Size TLB，单一页大小TLB）。

缺省状态下， LA32的整个虚拟地址空间配置如下：

区段名	地址范围	属性
UVRANGE	0x00000000 - 0x7FFFFFFF	分页映射，可缓存，PLV0~3
KPRANG	0x80000000 -	直接映射，非缓存，PLV0
KPRANGE1	0xA0000000 - 0xBFFFFFFF	直接映射，可缓存，PLV0
KVRAN	0xC0000000 -	分页映射，可缓存，PLV0

用 户 态 （ PLV3 ） 只 能 访 问 UVRANGE， 对 于 直 接 映 射 的 KPRANGE0 和 KPRANGE1， 将 虚 拟 地 址 的 第 30~31位清零就等于物理地址。例如：物理地址0x00001000对应的非缓存直接映射虚拟地址 是
0x80001000，而其可缓存直接映射虚拟地址是0xA0001000。

缺省状态下， LA64的整个虚拟地址空间配置如下：

区段名	地址范围	属性
XUVRANGE	0x0000000000000000 - 0x3FFFFFFFFFFFFFFF	分页映射，可缓存，PLV0~3
XSPRAN	0x40000000000000 00 - 0x7FFFFFFFFFFFFFFF	直接映射，可缓存 / 非缓存，PLV0
XKPRANGE	0x8000000000000000 - 0xBFFFFFFFFFFFFFFF	直接映射，可缓存 / 非缓存，PLV0
XKVRAN	0xC0000000000000 00 - 0xFFFFFFFFFFFFFFFF	分页映射，可缓存，PLV0

用户态（PLV3）只能访问XUVRANGE，对于直接映射的XSPRANGE和XKPRANGE，将虚拟地址的第 60~63位清零就等于物理地址，而其缓存属性是通过虚拟地址的第 60~61位配置的（0表示强序 非缓 存， 1表示一致可缓存， 2表示弱序非缓存）。

目前，我们仅用XKPRANGE来进行直接映射， XSPRANGE保留给以后用。

此处给出一个直接映射的例子：物理地址 0x00000000_00001000的强序非缓存直接映射虚拟地址（在 XKPRANGE中）是 0x80000000_00001000，其一致可缓存直接映射虚拟地址（在 XKPRANGE中） 是 0x90000000_00001000，而其弱序非缓存直接映射虚拟地址（在 XKPRANGE中）是0xA0000000_
00001000

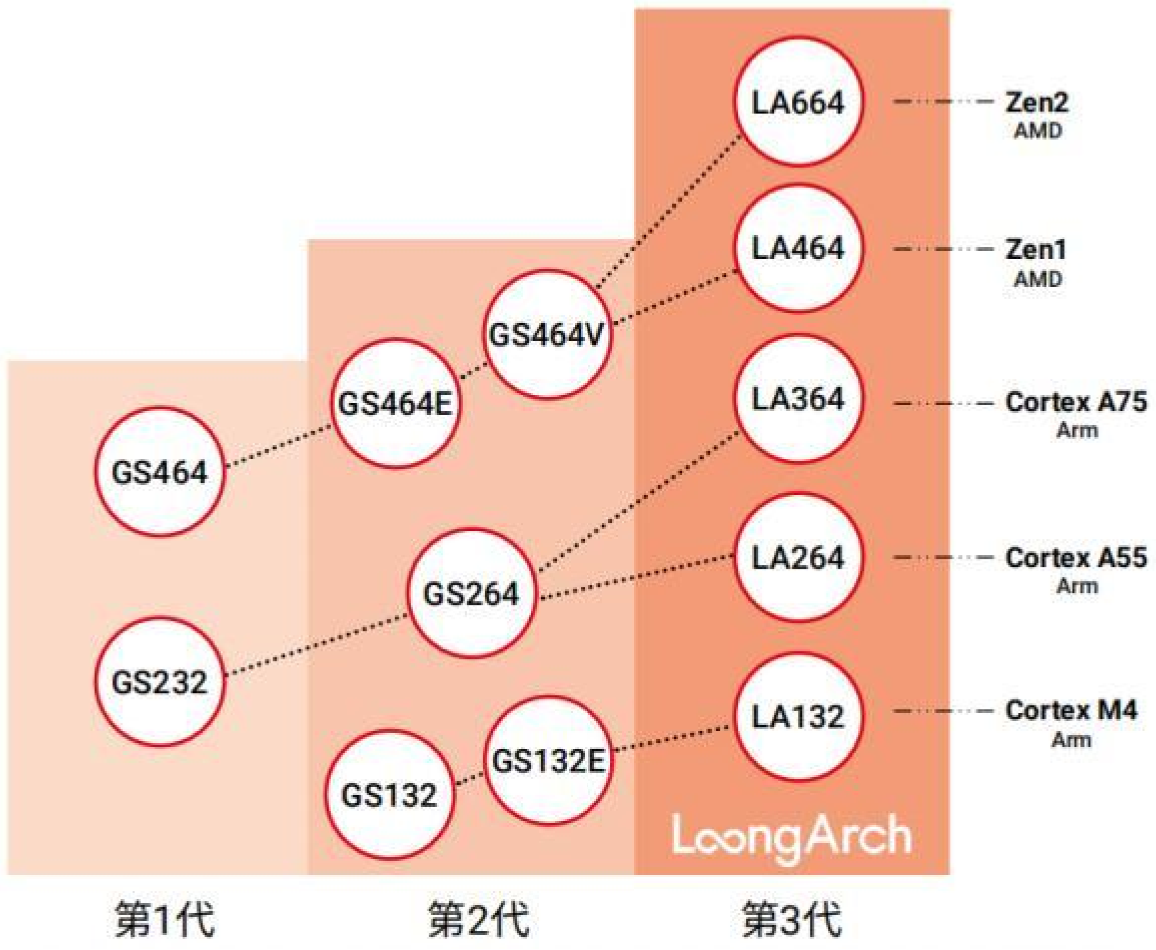
1.4 Loongson与LoongArch的关系

LoongArch是一种RISC指令集架构（ISA），不同于现存的任何一种ISA，而Loongson（即龙芯）是一 个处理器家族。

龙芯包括三个系列： Loongson-1（龙芯1号）是32位处理器系列， Loongson-2（龙芯2号）是低端64 位处理器系列，而Loongson-3（龙芯3号）是高端64位处理 器系列。

旧的龙芯处理器兼容MIPS架构，而新的龙芯处理器基于LoongArch架构。以龙芯3号为例：龙芯 3A1000/3B1500/3A2000/3A3000/3A4000都是兼容MIPS的，而龙芯
3A5000/3B5000/3C5000/3D5000/3C6000/3D6000 （以及将来的型号）都是基于LoongArch的。龙芯2
号系列，龙芯2K3000/2K2000/2K1500/2K1000/2K0500/2K0300 （以及将来的型号）也都是基于 LoongArch的。

1.5 龙芯系列处理器核(LoongArch架构)



龙芯基于LoongArch研发了LA132、LA264、LA364、LA464和LA664五大系列处理器核。

LA132为单发射32位结构，采用静态流水线，性能对标Cortex M4，在龙芯1C102、1C103等MCU中使 用。

LA264为双发射32/64位结构，采用动态流水线，性能对标*Cortex A55*，在龙芯2K0500
、2K1000LA、 2K0300等*SOC*中使用。

LA364为三发射64位结构，采用动态流水线，性能对标*Cortex A75*，在龙芯2K2000等*SOC*
中使用。

LA464为四发射64位结构，采用动态流水线， *SPECCPU2006*分值为10-12分/GHz。双访存、四定点、双 向量、 128项重排序缓存。性能对标*Zen1*，在龙芯3A5000、3C5000、
3D5000等*CPU*中使用。

LA664为六发射64位结构，采用动态流水线， *SPECCPU2006*分值为14-17分/GHz。四访存
、四定点、四
向量、两路同时多线程（SMT2）、256项重排序缓存。性能对标*Zen2*，在龙芯3A6000、
3C6000、 3D6000等*CPU*中使用。

LA132及LA264系列CPU核将开放给合作伙伴。
LA364核可以用于对战略客户的IP授权及SOC设计服务。
LA464、 LA664系列CPU核限于自用。

1.6 参考文献

Loongson官方网站（龙芯中科技术股份有限公司）：

> http://www.loongson.cn/

Loongson与LoongArch的开发者网站（软件与文档资源）：

> http://www.loongnix.cn/
>
> https://github.com/loo
>
> ngson/

https://github.com/loo

ngsonlab

https://loongson.github.io/LoongArch-Documentation/

LoongArch指令集架构的文档：

> https://github.com/loongson/LoongArch-
> Documentation/releases/download/2023.04.20/Lo ongArch-Vol1-
> v1.10-CN.pdf（中文版）

https://github.com/loongson/LoongArch-
Documentation/releases/download/2023.04.20/Lo ongArch-Vol1-
v1.10-EN.pdf（英文版）

LoongArch的ELF psABI文档：

> https://github.com/loongson/LoongArch-
> Documentation/releases/download/2023.04.20/Lo ongArch-ELF-
> ABI-v2.01-CN.pdf（中文版）

https://github.com/loongson/LoongArch-
Documentation/releases/download/2023.04.20/Lo ongArch-ELF-
ABI-v2.01-EN.pdf （英文版）

LoongArch工具链规约文档：

> https://github.com/loongson/LoongArch-Documentation/releases/download/2023.04.20/LoongArch-toolchain-conventions-v1.00-CN.pdf（中文版）
>
> https://github.com/loongson/LoongArch-Documentation/releases/download/2023.04.20/LoongArch-toolchain-conventions-v1.00-EN.pdf（英文版）

Loongson与LoongArch的Linux内核源码仓库：

> https://git.kernel.org/pub/scm/linux/kernel/git/chenhuacai/linux-loongson.git
>
> https://github.com/chenhuacai/linux/tree/loongarch-next

二、开发板简介

1、概述

龙芯 2K0300 先锋派是广东龙芯基于龙芯自研 LoongArch 架构 64 位 SoC 处理器 2K0300 设计的单板方 案，板卡尺寸为(85mm x 56mm)，兼容树莓派 4B 尺寸大小、定位孔及 40 PIN GPIO 定义。板卡接口资 源丰富，外设生态扩展方便，支持多种操作系统及图形 GUI 开发设计，资料配套齐全。板卡采用全表贴 化设计，核心元器件均可采用国产器件替换，具有自主、安全、稳定、可靠、实用性强等特点，可广泛 用于工业自动化控制、工业网关，物联网数采、能源电力、智慧水务、轨道交通、教学教具等应用领域 的方案学习评估和技术预研。

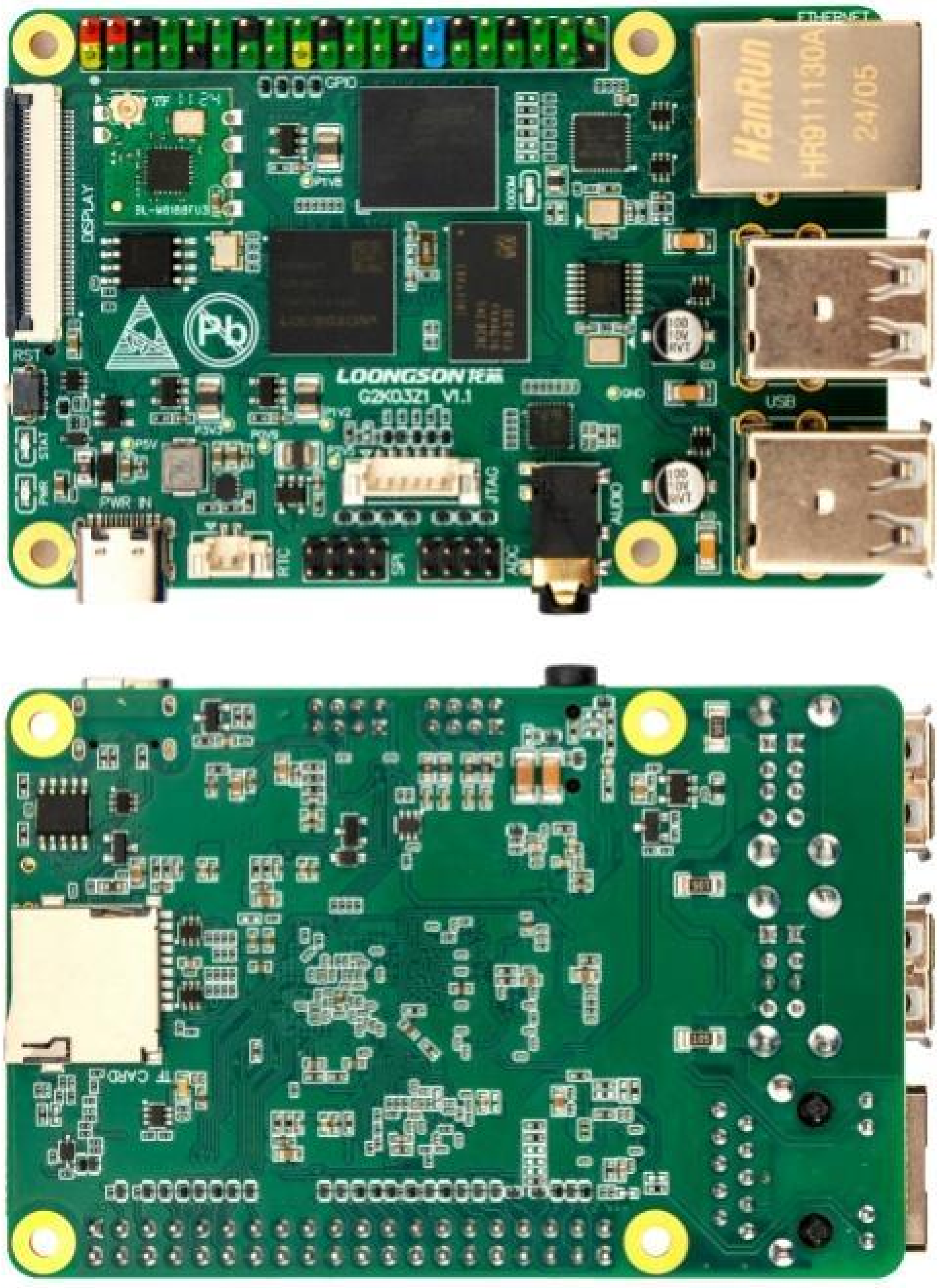


图1-1广东龙芯2K0300先锋派实物图

2、硬件规格

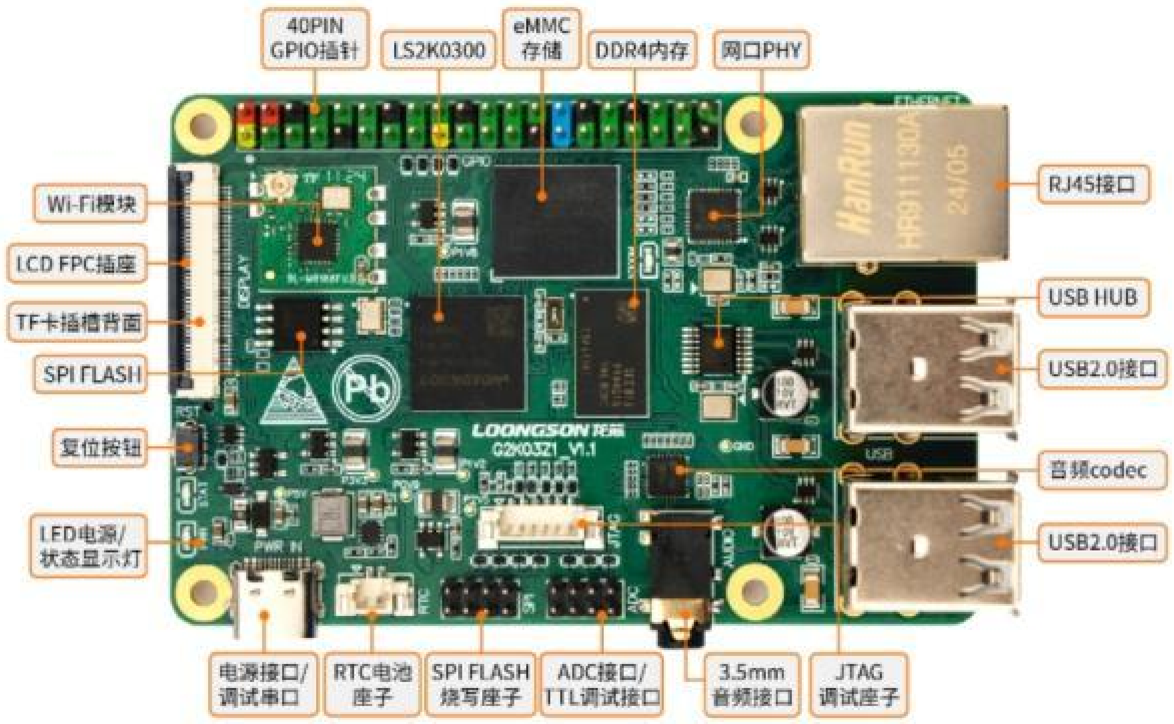


图1-2 广东龙芯2K0300先锋派接口图

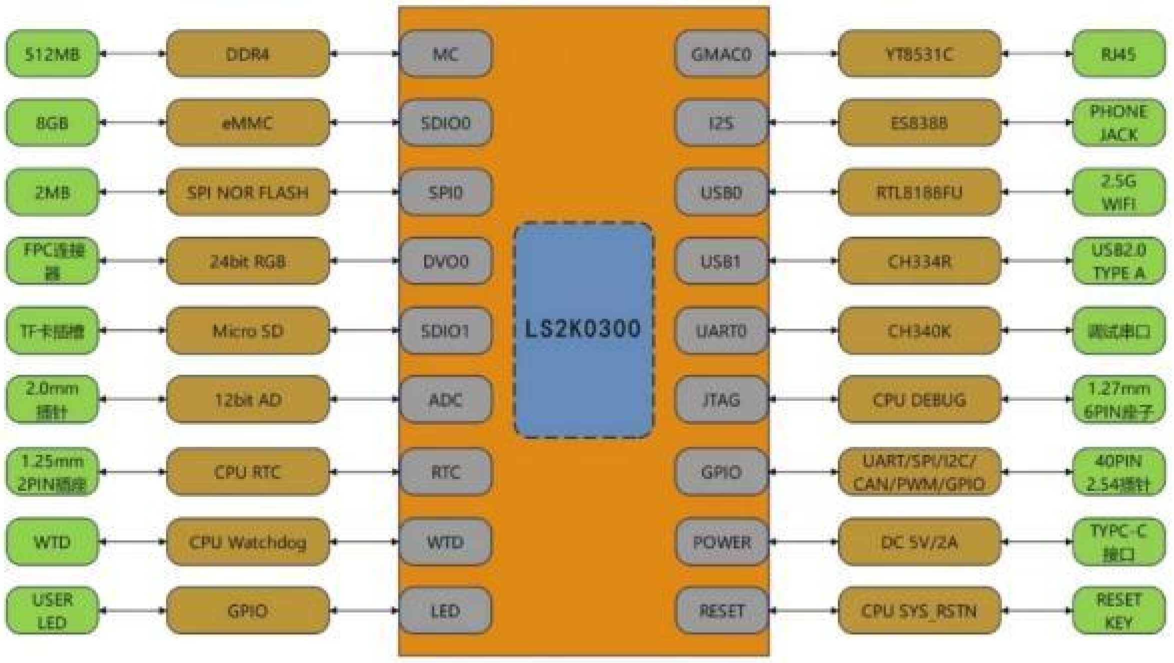
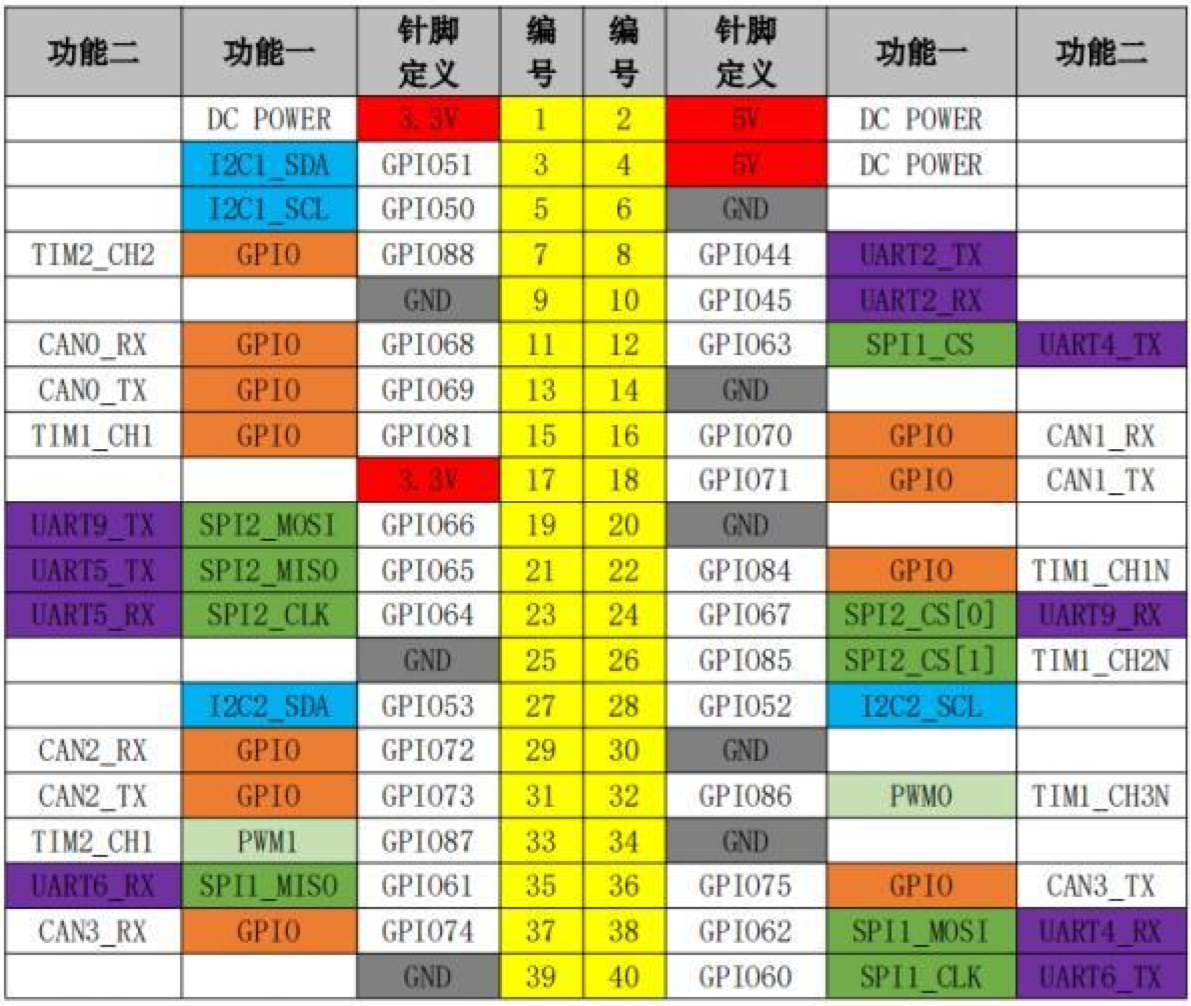


图1-3 广东龙芯2K0300先锋派功能图

40 PIN GPIO 采用2.54mm 插针形式引出，接口复用丰富 ，可复用为 UART/I2C/SPI/CAN/PWM/TIM/GPIO 等接口，可兼容树莓派接口，复用树莓派外围 扩展硬件模块。为此我们定义**功能一**和**功能二**两种复用功能。



功能二	功能一	针脚定义	编号	编号	针脚定义	功能一	功能二
	DC POWER	3.3V	1	2	5V	DC POWER	
	I2C1_SDA	GPIO51	3	4	5V	DC POWER	
	I2C1_SCL	GPIO50	5	6	GND		
TIM2_CH2	GPIO	GPIO88	7	8	GPIO44	UART2_TX	
		GND	9	10	GPIO45	UART2_RX	
CAN0_RX	GPIO	GPIO68	11	12	GPIO63	SPI1_CS	UART4_TX
CAN0_TX	GPIO	GPIO69	13	14	GND		
TIM1_CH1	GPIO	GPIO81	15	16	GPIO70	GPIO	CAN1_RX
		3.3V	17	18	GPIO71	GPIO	CAN1_TX
UART9_TX	SPI2_MOSI	GPIO66	19	20	GND		
UART5_TX	SPI2_MISO	GPIO65	21	22	GPIO84	GPIO	TIM1_CH1N
UART5_RX	SPI2_CLK	GPIO64	23	24	GPIO67	SPI2_CS[0]	UART9_RX
		GND	25	26	GPIO85	SPI2_CS[1]	TIM1_CH2N
	I2C2_SDA	GPIO53	27	28	GPIO52	I2C2_SCL	
CAN2_RX	GPIO	GPIO72	29	30	GND		
CAN2_TX	GPIO	GPIO73	31	32	GPIO86	PWM0	TIM1_CH3N
TIM2_CH1	PWM1	GPIO87	33	34	GND		
UART6_RX	SPI1_MISO	GPIO61	35	36	GPIO75	GPIO	CAN3_TX
CAN3_RX	GPIO	GPIO74	37	38	GPIO62	SPI1_MOSI	UART4_RX
		GND	39	40	GPIO60	SPI1_CLK	UART6_TX

图1-4 广东龙芯2K0300先锋派40PIN引脚复用图

表1-1广东龙芯2K0300先锋派接口描述

编号	板载硬件资源	描述
1	CPU	LOONGSON 2K0300处理器，主频1GHz 64位 SoC LA264
2	内存	16 位 DDR4 控制器，容量512MB
3	NOR FLASH	容量2MByte,用于烧录启动系统程序
4	EMMC	容量8GByte,EMMC支持4/8线
5	USB Type -C接 口	USB转UART接口，方便用户调试；同时给整板供电
6	JTAG调试	JTAG用于测试调试
7	TTL调试串口	1*UART 调试接口，当负载比较重时type-c只负责供电，可用此接口 做串口调试
8	RTC电池座	安装CR2032纽扣电池，供RTC使用，为开发板提供精确时间
9	LCD接 口	可支持24bit LCD显示屏
10	TF卡座	支持自弹式TF卡，用于存储操作系统镜像和文件系统

编号	板载硬件资源	描述
11	LED指示灯	1个用户自定义指示灯
12	复位按键	开发板复位按键
13	Audio	1*3.5mm 音频接口，支持录音播放
14	千兆以太网接口	1路千兆以太网接口，用于和电脑或其他网络设备进行以太网数据交换
15	USB HOST接口	4路USB 2.0 HOST接口
16	GPIO	1*40 PIN GPIO（可复用5路UART/2路I2C/3路SPI/4路 CAN/4 路 PWM）
17	AD模拟输入接 口	1*12 位 ADC 接口，引出 4 通道
18	SPI烧录接口	用于烧录SPI NOR FLASH芯片的接口
19	电源指示灯	用于指示板卡供电电源是否正常

3、尺寸大小

2K0300先锋派尺寸为： 85mm*56mm，如图所示。

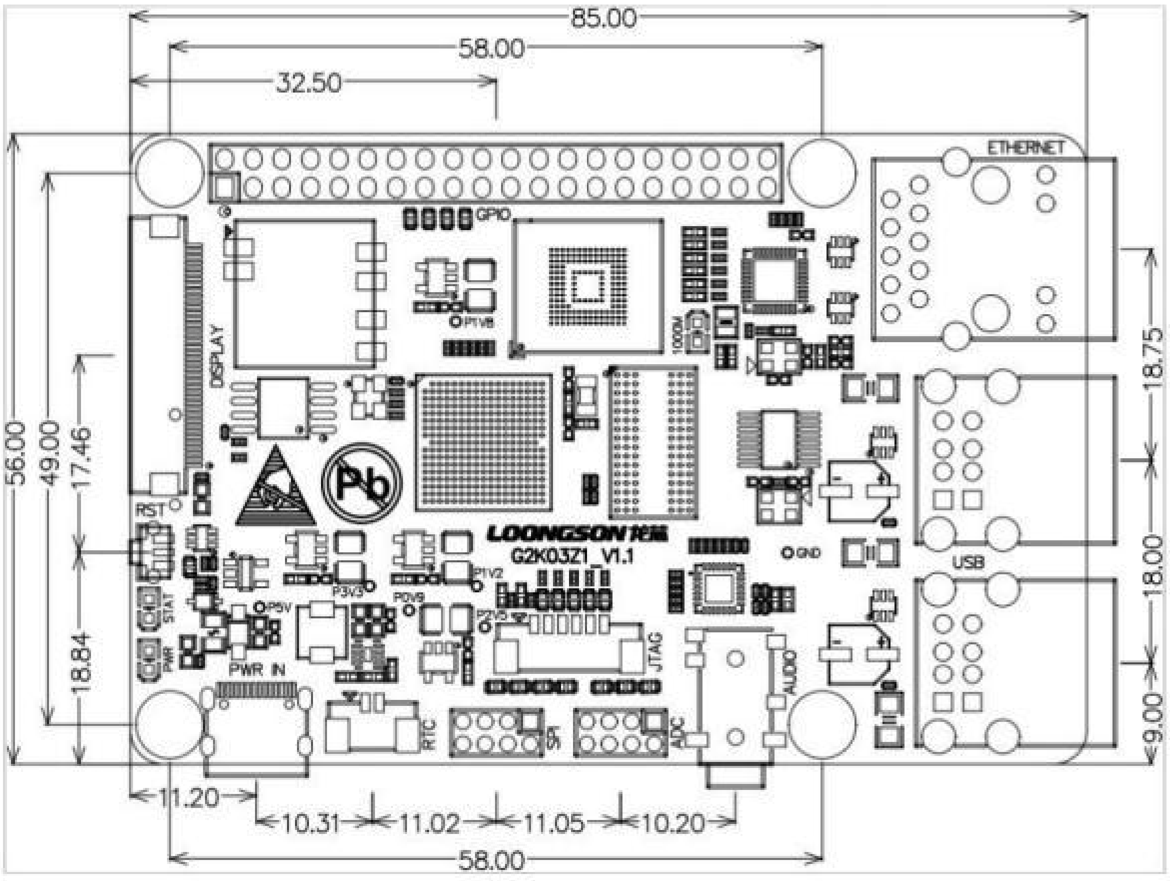


图1-5 广东龙芯2K0300先锋派尺寸图

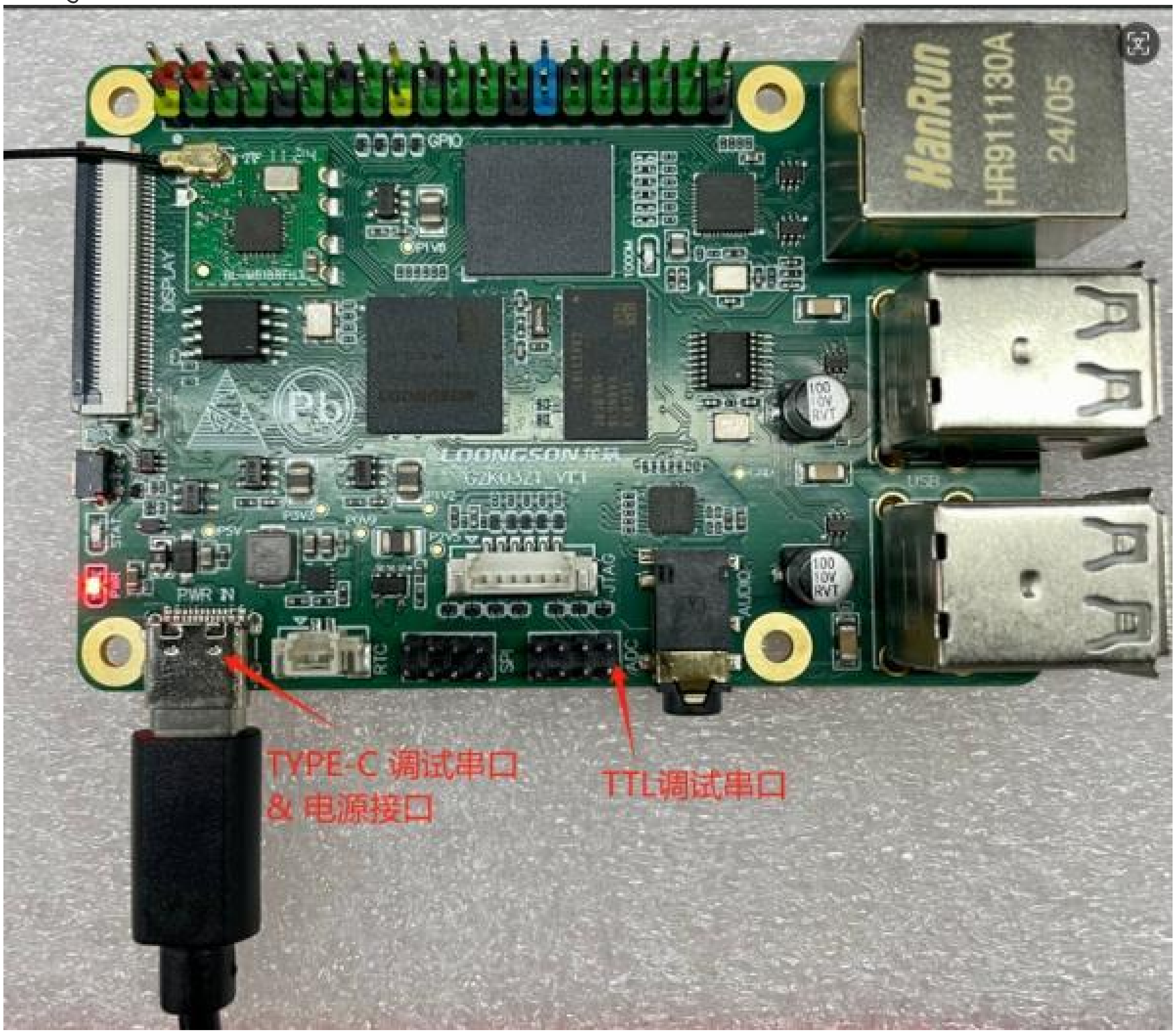
四、开发板上手

4.1.先显示出来

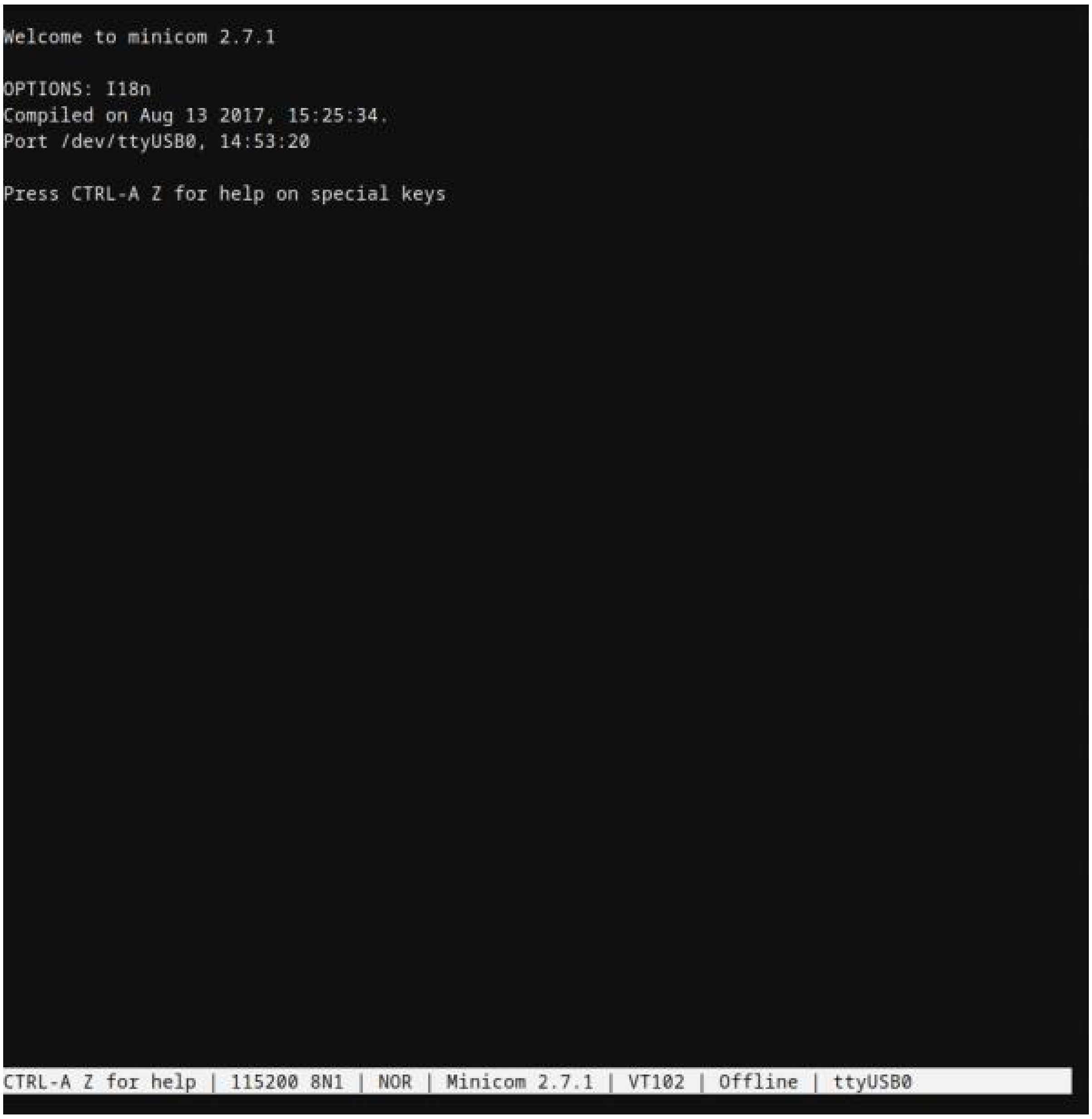
4.1.1.使用串口调试

1.UARTO 做为 Debug 串口，配套 TYPE-C 线一端接板卡，另一端接PC。**TYPE-C线同时可以做为电源供 电，如果板卡上的负载较大时，建议TYPE-C USB-A接口接在5v电源适配器上，然后使用网络或TTL调试 串口进行调试。**

Debug 串口接线如下图：



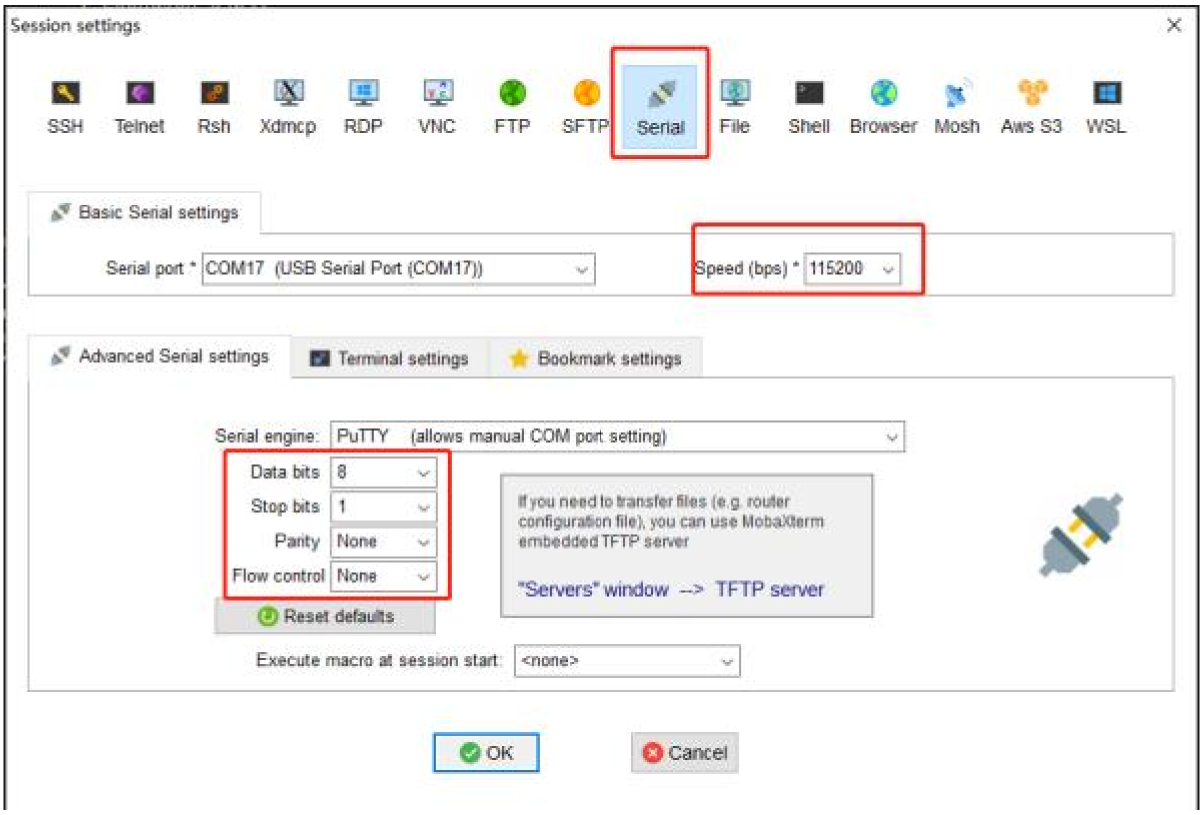
2.PC上打开串口工具（比如windows MobaXterm ，linux minicom 等），配置好串口（**选择好串口 号，设置波特率：115200，数据位：8，停止位：1，硬件流控：无**），等待上电。



```
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB0, 14:53:20

Press CTRL-A Z for help on special keys
```

```
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB0
```

MobaXterm的 串 口 设 置 图



Serial port * COM17 (USB Serial Port (COM17))
Speed (bps) * 115200

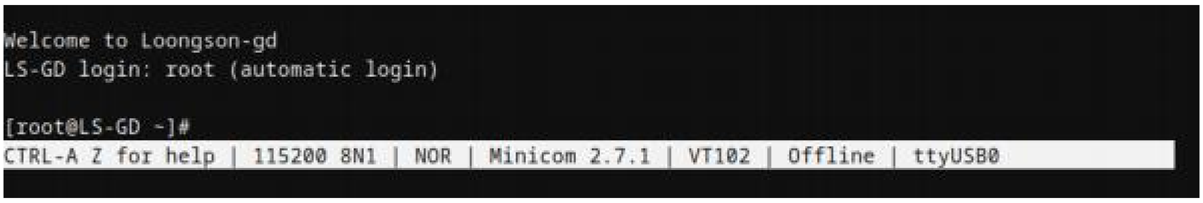
Serial engine: PuTTY (allows manual COM port setting)
Data bits 8
Stop bits 1
Parity None
Flow control None

If you need to transfer files (e.g. router configuration file), you can use MobaXterm embedded TFTP server

"Servers" window --> TFTP server

Execute macro at session start: <none>

3.按下开发板上POWER键启动，并进入预置Busybox系统，默认自动登录； **系统默认账户为** root **，默 认密码为** 123**，默认IP为192.168.1.10**。



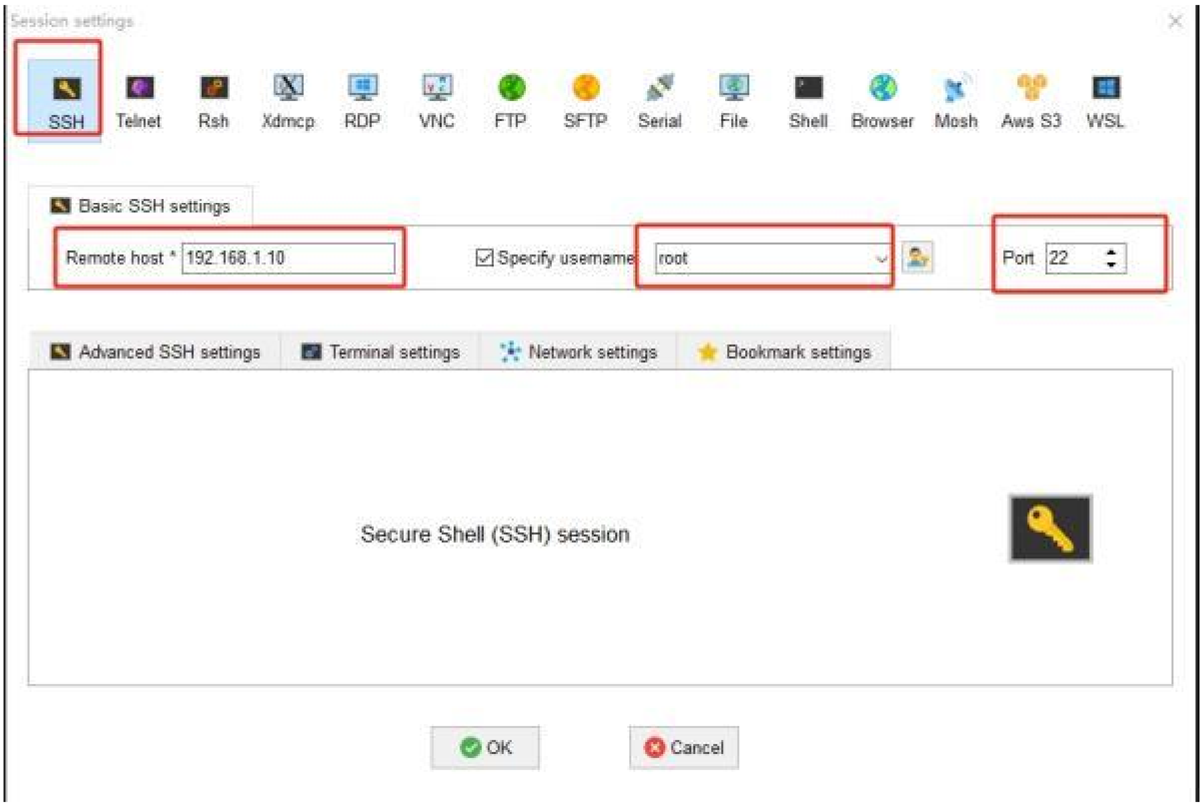
```
Welcome to Loongson-gd
LS-GD login: root (automatic login)

[root@LS-GD ~]#
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB0
```

4.1.2.使用网口调试

网盘里面的文件系统均带有ssh服务，可以通过ssh 登录到板卡上进行调试。将板卡与电脑通过网线直 连，同时配置电脑的IP 为 192.168.1.2，也可以是同一网段的其他IP，建议配成 192.168.1.2，因为 u-boot 中默认配的serverip 是 192.168.1.2。**系统默认账户为** root**，默认密码为** 123**，默认IP为**
192.168.1.10，SSH默认端口为： 22。

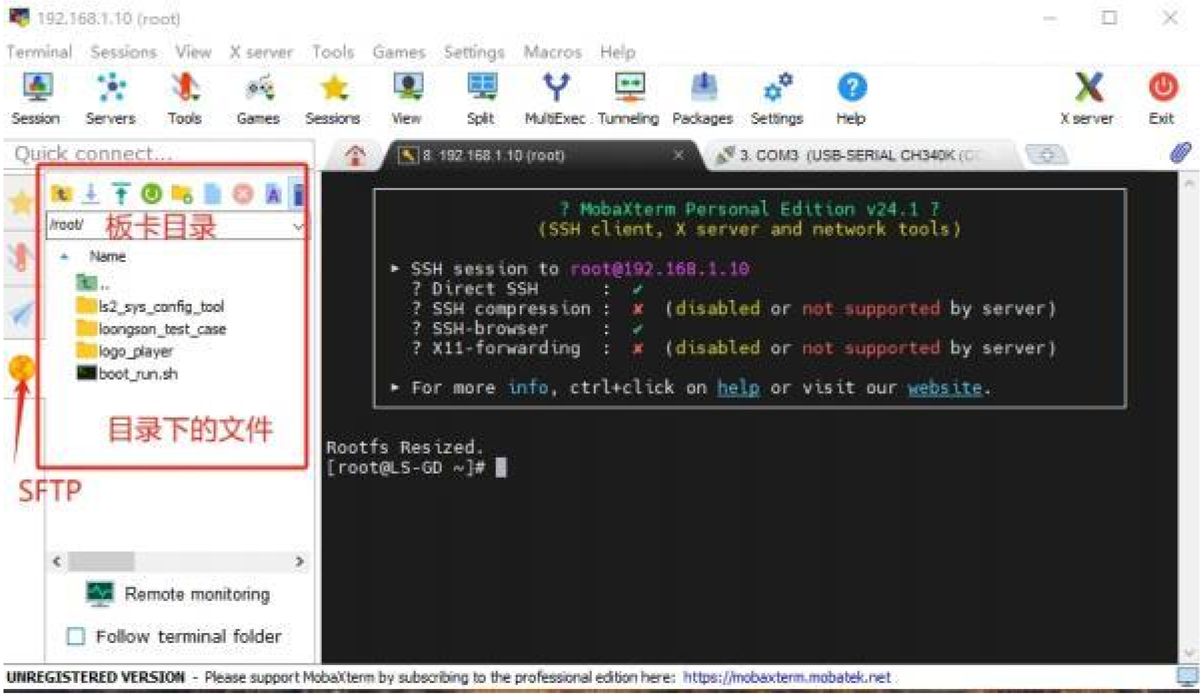
> 如出现板卡Ping 不通电脑，但电脑可以Ping 通板卡的现象，请检查电脑的防火墙状态，将其关闭 后再试。

MobaXterm的SSH设置图：



Remote host * 192.168.1.10
☑ Specify username root
Port 22

ssh 连接的后图如下：


192.168.1.10 (root)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
8. 192.168.1.10 (root)
3. COM3 (USB-SERIAL CH340K (C
/root/
板卡目录
Name
..
ls2_sys_config_tool
loongson_test_case
logo_player
boot_run.sh
目录下的文件
SFTP
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```
          ? MobaXterm Personal Edition v24.1 ?
        (SSH client, X server and network tools)

 ▸ SSH session to root@192.168.1.10
   ? Direct SSH      :  ✔
   ? SSH compression :  ✘  (disabled or not supported by server)
   ? SSH-browser     :  ✔
   ? X11-forwarding  :  ✘  (disabled or not supported by server)

 ▸ For more info, ctrl+click on help or visit our website.

Rootfs Resized.
[root@LS-GD ~]# 
```

ssh 连接后可以通过SFTP 传输文件，支持拖拽式进行文件上下传。

4.1.3.使用LCD

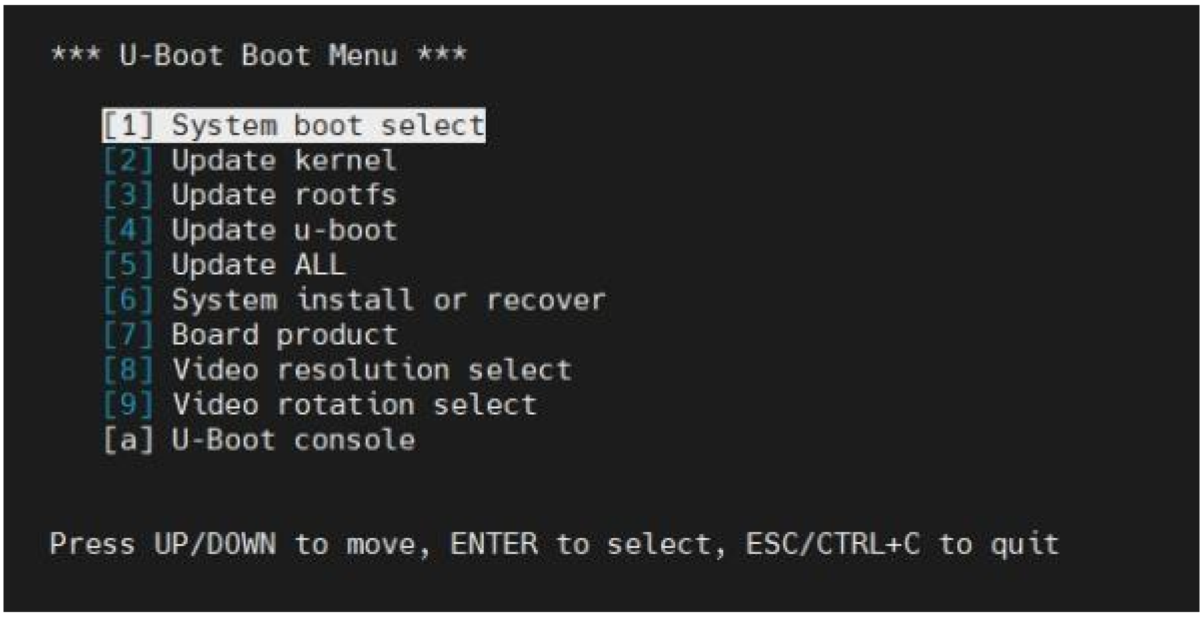
我们适配了多种LCD屏的分辨率(800x480，1024x600，1280x800)，覆盖了4.3'，7'，10.1'等主流尺寸的 LCD屏，默认分辨率为7' 1024x600，分辨率切换参考4.2.2.切换LCD分辨率 。

LCD 接口接线如下图：

4.2.首次进入 uboot

蜂鸟板的文件系统是典型的linux系统，一些操作事项参考 五、文件系统。本章之后的内容我们将着重说 明在 uboot 中的操作办法，让我们先进入 uboot 菜单：

开机按住“m”键进入u-boot菜单



```
*** U-Boot Boot Menu ***

    [1] System boot select
    [2] Update kernel
    [3] Update rootfs
    [4] Update u-boot
    [5] Update ALL
    [6] System install or recover
    [7] Board product
    [8] Video resolution select
    [9] Video rotation select
    [a] U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

在u-boot 菜单中选择 “[a] U-Boot console”进入u-boot 命令终端 或者开机按“c”进入 u-boot console

在u-boot 命令终端输入 `bootmenu` 命令也可以进入u-boot 菜单

```
=> bootmenu
  *** U-Boot Boot Menu ***

      [1] System boot select
      [2] Update kernel
      [3] Update rootfs
      [4] Update u-boot
      [5] Update ALL
      [6] System install or recover
      [7] Board product
      [8] Video resolution select
      [9] Video rotation select [a]
      U-Boot console



  Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

在命令终端中键入 `boot`，即可进入正常启动流程

```
=> boot
--if mmc0 --fmt ext4 --sym /boot/uImage
--of ram --fmt  --sym
--extra 0x0
loading...
loaded&burning 8812660 bytes ...
load&burn 8812660 finished
## Booting kernel from Legacy Image at 9000000003000000 ...
   Image Name:   Linux-5.10.0.lsgd-ga653fe8758c4
   Image Type:   LoongArch Linux Kernel Image (gzip compressed)
   Data Size:    8812596 Bytes = 8.4 MiB
   Load Address: 00200000
   Entry Point:  00e20250
   Verifying Checksum ... OK
   Uncompressing Kernel Image
Warning: invalid device tree. Used linux default dtb
512 MiB
```

```
[    0.000000] Linux version 5.10.0.lsgd- ...
```

在命令终端中键入`reboot/` `reset`，即可重启

```
=> reboot
LoongArch Initializing ...
```

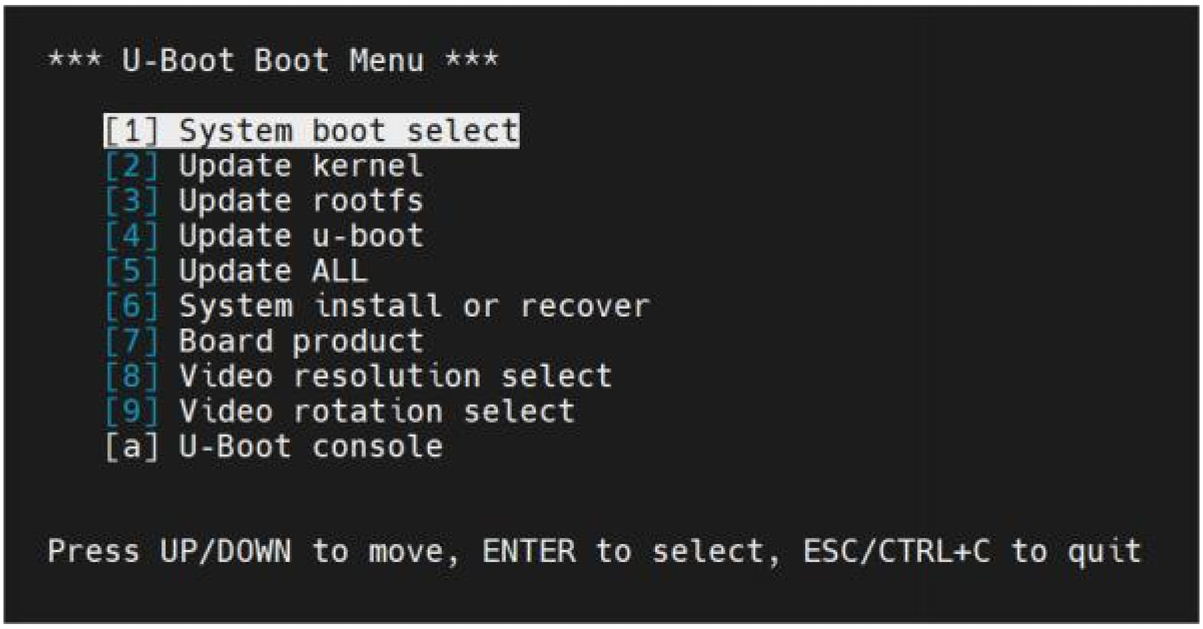
```
RAM(Cache AS RAM) Initializing ... Lock


Scache Done.


Copy spl code to locked scache...


Jump to board_init_f...

Enter board_init_f...


 -   _  _ - - __ __ _ _ -   /  __ _ \
 |   | || |N| |_[_ | |N|  | |_| \|
 |_ |_| |_| |\| |_]__] |_| |\|   \ |_] |_/ /
 ============ddr4 init and training done!========
Trying to boot from BootSpace U

-boot start ...

Jump to board_init_f...

U-Boot 2022.04-v2.1.0-00513-g095bcbdd (Aug 20 2024 - 14:33:16 +0800), Build:
```

下面我们介绍几个比较简单的菜单项

4.2.1.启动模式选择

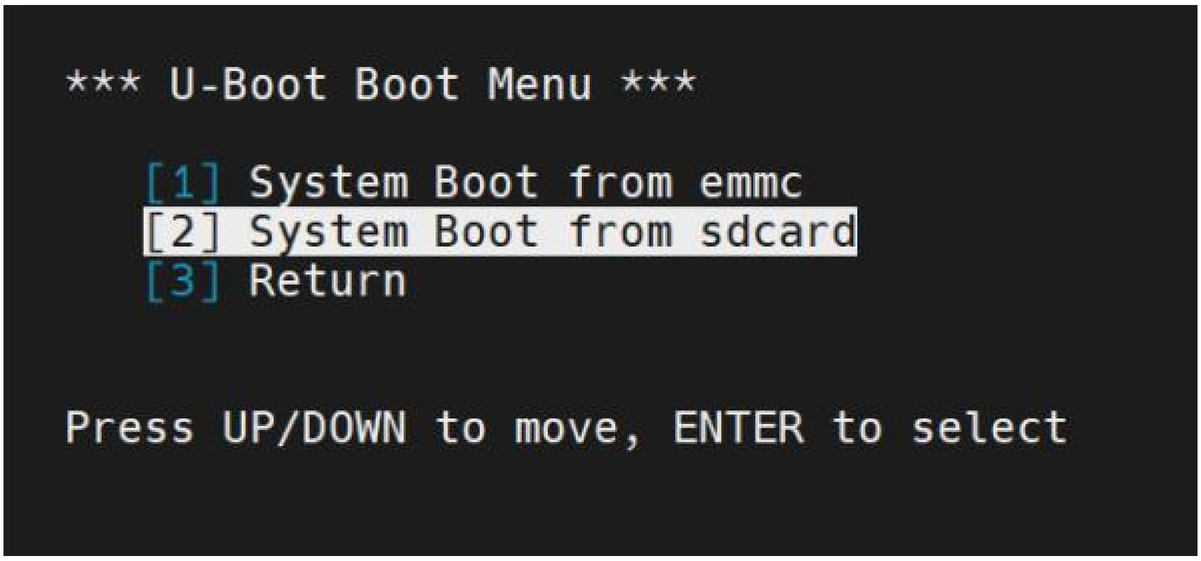
蜂鸟板支持从多种介质启动系统，前提是该介质安装有系统，默认从EMMC启动。可以按以下流程更改 启动介质。

1.uboot菜单选择“[1] System boot select”



```
*** U-Boot Boot Menu ***

    [1] System boot select
    [2] Update kernel
    [3] Update rootfs
    [4] Update u-boot
    [5] Update ALL
    [6] System install or recover
    [7] Board product
    [8] Video resolution select
    [9] Video rotation select
    [a] U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

2.选择“[1] System Boot from emmc” 或 “[2] System Boot from sdcard”

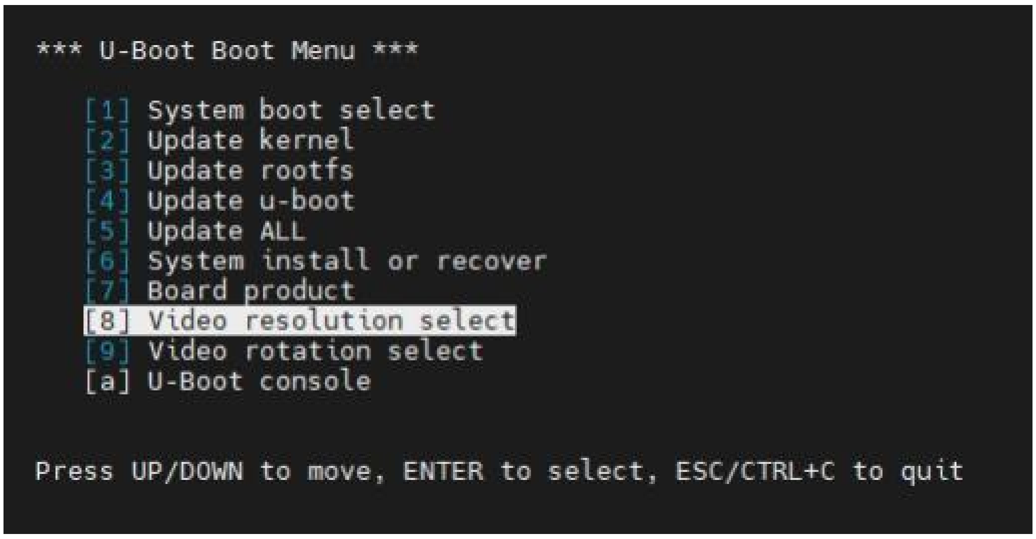


```
*** U-Boot Boot Menu ***

    [1] System Boot from emmc
    [2] System Boot from sdcard
    [3] Return


Press UP/DOWN to move, ENTER to select
```

4.2.2.切换LCD分辨率

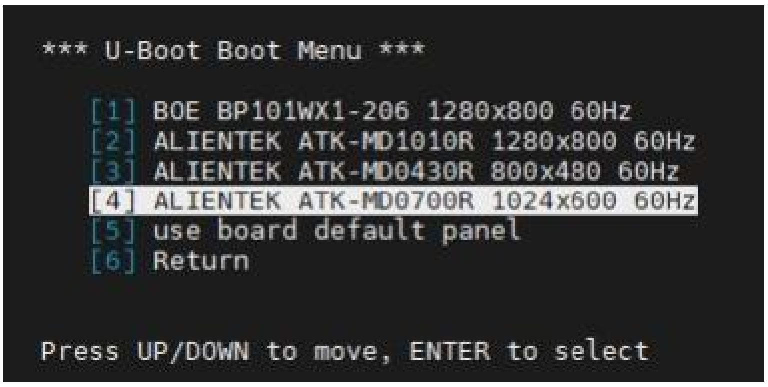
> 如果切换之后，显示花屏，可检查是否使用了新的u-boot 和 kernel ，以及板卡供电是否满足。

1.uboot菜单选择“[8] Video resolution select”



```
*** U-Boot Boot Menu ***

    [1] System boot select
    [2] Update kernel
    [3] Update rootfs
    [4] Update u-boot
    [5] Update ALL
    [6] System install or recover
    [7] Board product
    [8] Video resolution select
    [9] Video rotation select
    [a] U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

2.选择相应的分辨率



```
*** U-Boot Boot Menu ***

    [1] BOE BP101WX1-206 1280x800 60Hz
    [2] ALIENTEK ATK-MD1010R 1280x800 60Hz
    [3] ALIENTEK ATK-MD0430R 800x480 60Hz
    [4] ALIENTEK ATK-MD0700R 1024x600 60Hz
    [5] use board default panel
    [6] Return


Press UP/DOWN to move, ENTER to select
```

4.3.uboot下常用外设与网络服务

4.3.1. U盘-FAT32

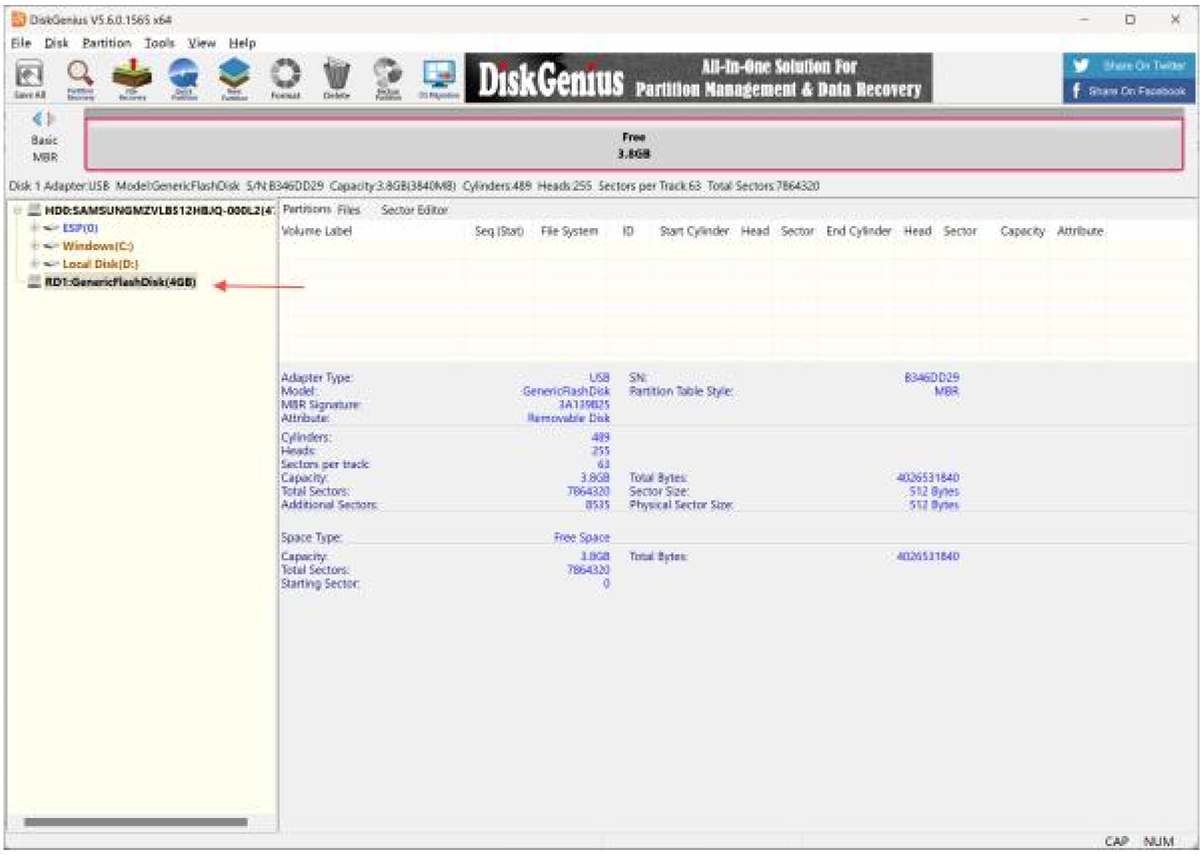
考虑到linux与Windows的兼容性，我们选择将U盘格式设为 FAT32。

4.3.1.1. U盘准备

1.U盘格式化为FAT32

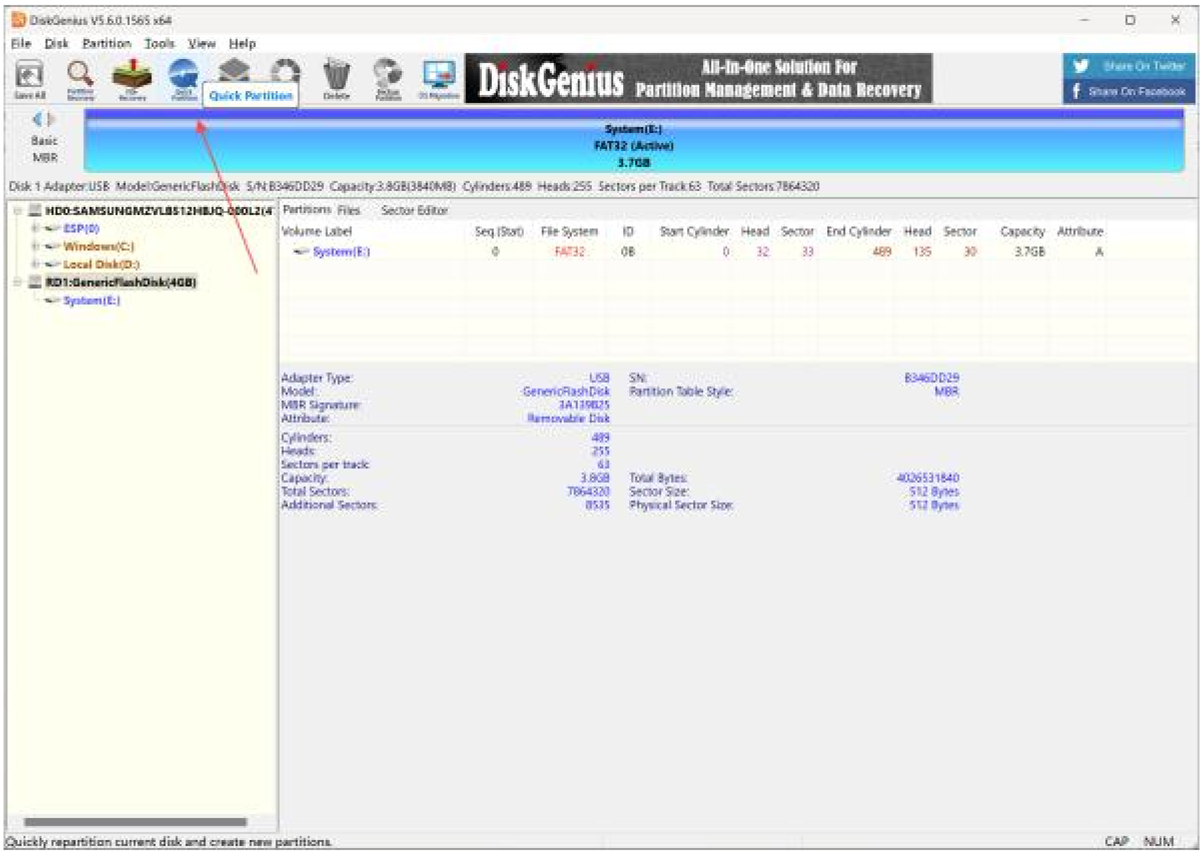
a.Windows 格式化U盘

推荐使用DiskGenius避免分区表错误。 **注意，请事先备份好U盘的数据**。演示请看下图： 插入U盘，打开DiskGenius，选中U盘设备


DiskGenius V5.6.0.1565 x64
File Disk Partition Tools View Help
Save All
Format
Delete
DiskGenius All-In-One Solution For Partition Management & Data Recovery
Share On Twitter
Share On Facebook
Basic
MBR
Free
3.8GB
HD0:SAMSUNGMZVLB512HBJQ-000L2(4
ESP(0)
Windows(C:)
Local Disk(D:)
RD1:GenericFlashDisk(4GB)
Partitions Files Sector Editor
Volume Label Seq (Stat) File System ID Start Cylinder Head Sector End Cylinder Head Sector Capacity Attribute
CAP NUM

```
Disk 1 Adapter:USB  Model:GenericFlashDisk  S/N:B346DD29  Capacity:3.8GB(3840MB)  Cylinders:489  Heads:255  Sectors per Track:63  Total Sectors:7864320

Adapter Type:                 USB   SN:                         B346DD29
Model:            GenericFlashDisk   Partition Table Style:           MBR
MBR Signature:            3A139825
Attribute:          Removable Disk

Cylinders:                     489
Heads:                         255
Sectors per track:              63
Capacity:                    3.8GB   Total Bytes:              4026531840
Total Sectors:             7864320   Sector Size:               512 Bytes
Additional Sectors:           8535   Physical Sector Size:      512 Bytes

Space Type:             Free Space

Capacity:                    3.8GB   Total Bytes:              4026531840
Total Sectors:             7864320
Starting Sector:                 0
```

点击“Quick Partition”


DiskGenius V5.6.0.1565 x64
File Disk Partition Tools View Help
Save All
Quick Partition
Delete
DiskGenius All-In-One Solution For Partition Management & Data Recovery
Share On Twitter
Share On Facebook
Basic
MBR
System(E:)
FAT32 (Active)
3.7GB
HD0:SAMSUNGMZVL8512HBJQ-00BL2(4
ESP(0)
Windows(C:)
Local Disk(D:)
RD1:GenericFlashDisk(4GB)
System(E:)
Partitions Files Sector Editor
CAP NUM

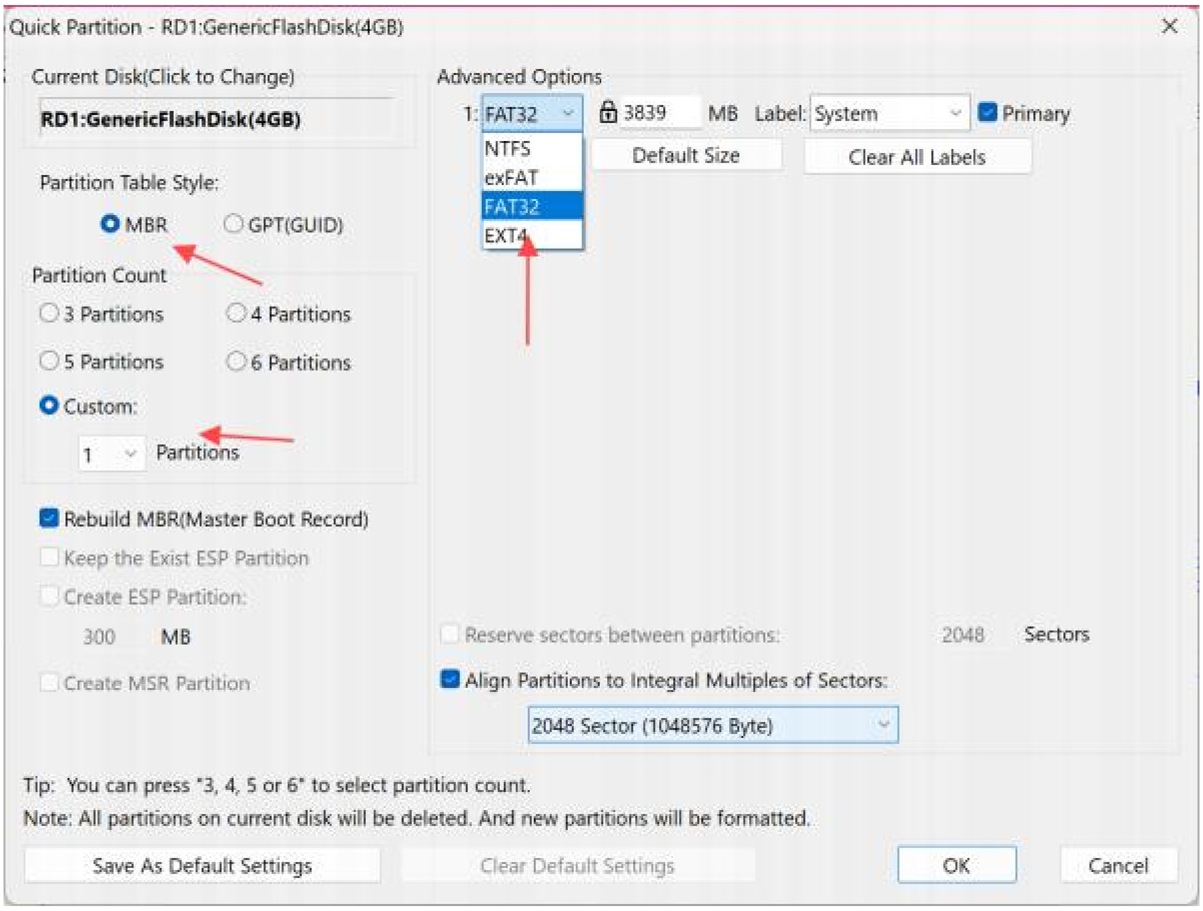
Disk 1 Adapter:USB Model:GenericFlashDisk S/N:B346DD29 Capacity:3.8GB(3840MB) Cylinders:489 Heads:255 Sectors per Track:63 Total Sectors:7864320

Volume Label	Seq (Stat)	File System	ID	Start Cylinder	Head	Sector	End Cylinder	Head	Sector	Capacity	Attribute
System(E:)	0	FAT32	0B	0	32	33	489	135	30	3.7GB	A

Adapter Type:	USB	SN:	B346DD29
Model:	GenericFlashDisk	Partition Table Style:	MBR
MBR Signature:	3A139B25		
Attribute:	Removable Disk		
Cylinders:	489		
Heads:	255		
Sectors per track:	63		
Capacity:	3.8GB	Total Bytes:	4026531840
Total Sectors:	7864320	Sector Size:	512 Bytes
Additional Sectors:	8535	Physical Sector Size:	512 Bytes

Quickly repartition current disk and create new partitions.

依次选MBR分区表、创建一个分区，格式fat32 ， Lable 随便填。点击“OK”开始分区



Current Disk(Click to Change)
RD1:GenericFlashDisk(4GB)

Partition Table Style:
[✓ MBR] GPT(GUID)

Partition Count
3 Partitions 4 Partitions
5 Partitions 6 Partitions
[✓ Custom:] 1 Partitions

[✓ Rebuild MBR(Master Boot Record)]
Keep the Exist ESP Partition
Create ESP Partition: 300 MB
Create MSR Partition

Advanced Options
1: FAT32 3839 MB Label: System [✓ Primary]

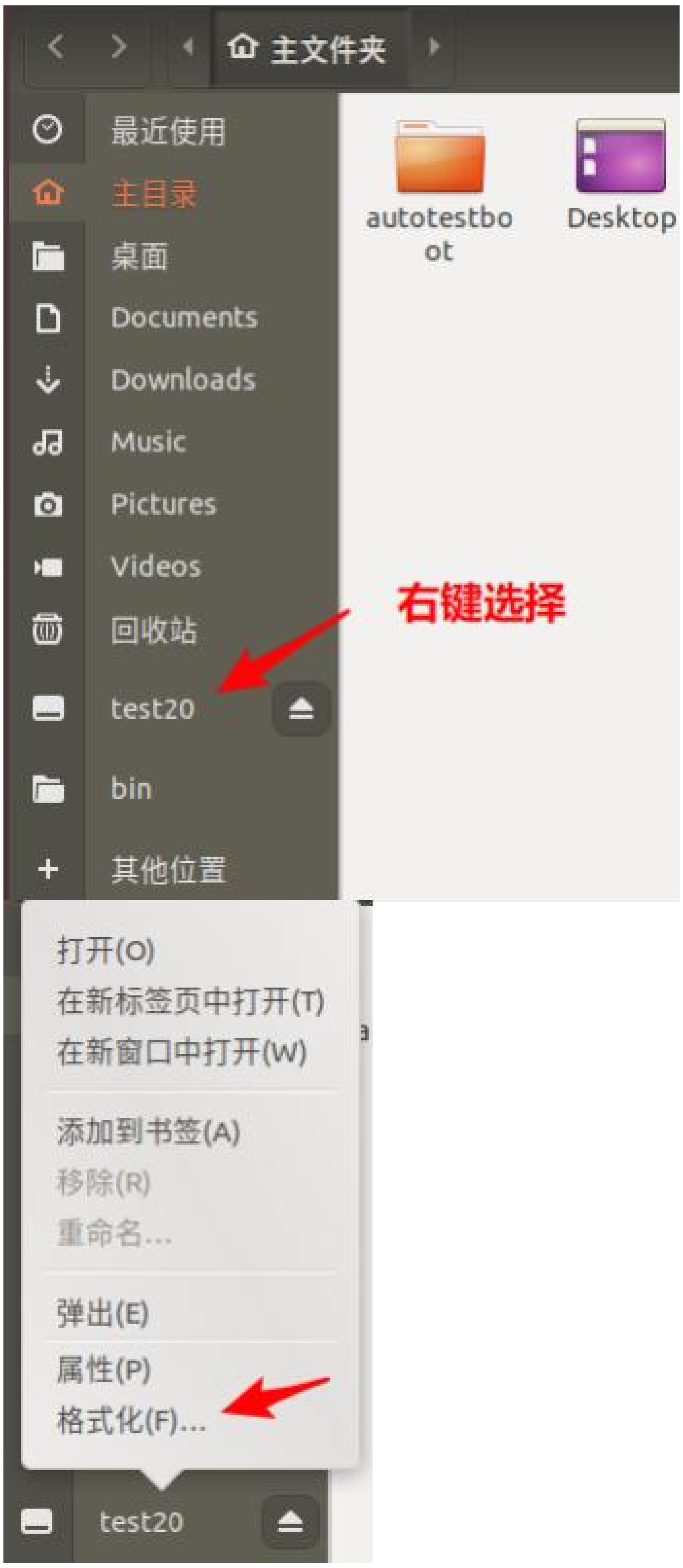
Reserve sectors between partitions: 2048 Sectors
[✓ Align Partitions to Integral Multiples of Sectors:]
2048 Sector (1048576 Byte)

Tip: You can press "3, 4, 5 or 6" to select partition count.
Note: All partitions on current disk will be deleted. And new partitions will be formatted.

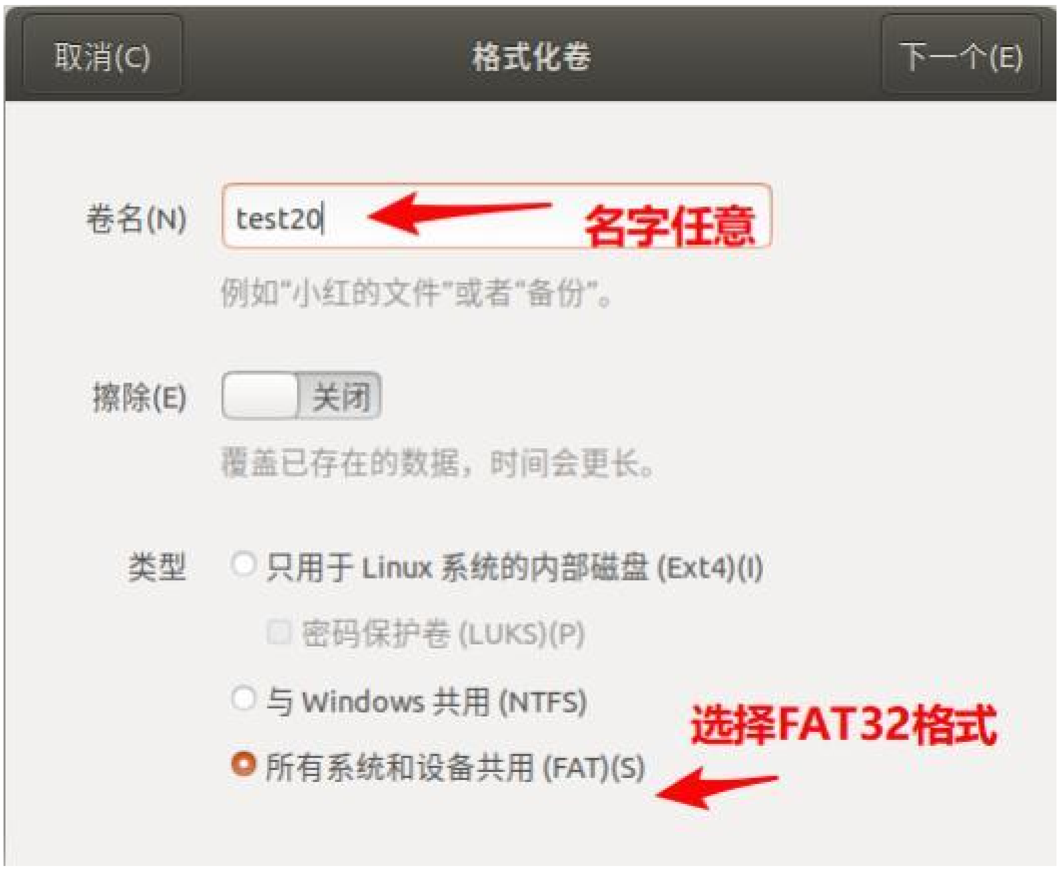
b. Linux 格式化U盘

推荐使用ubuntu18.04 （也就是推荐用于交叉编译开发的文件系统）的**自带U盘格式化工具。注意，请 事先备份好U盘的数据**。演示请看下图：

PC机插入U盘后，打开任意文件夹。


主文件夹
最近使用
主目录
桌面
Documents
Downloads
Music
Pictures
Videos
回收站
test20
bin
其他位置
autotestboot
Desktop
右键选择
打开(O)
在新标签页中打开(T)
在新窗口中打开(W)
添加到书签(A)
移除(R)
重命名…
弹出(E)
属性(P)
格式化(F)…
test20

然后点击“下一个”按钮。



卷名(N) test20

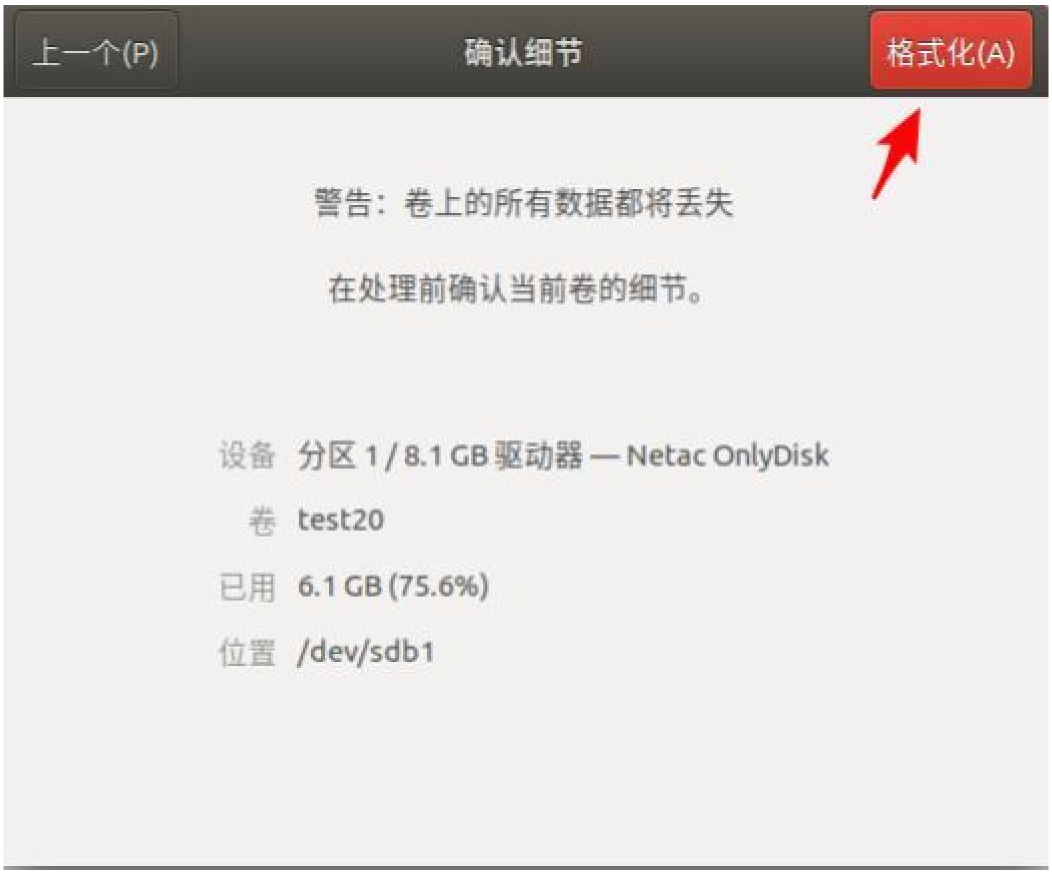
例如"小红的文件"或者"备份"。

擦除(E) 关闭

覆盖已存在的数据，时间会更长。

类型
- 只用于 Linux 系统的内部磁盘 (Ext4)(I)
- 密码保护卷 (LUKS)(P)
- 与 Windows 共用 (NTFS)
- [✓ 所有系统和设备共用 (FAT)(S)]

点击"格式化"按钮，即可等待完成格式化。



警告：卷上的所有数据都将丢失

在处理前确认当前卷的细节。

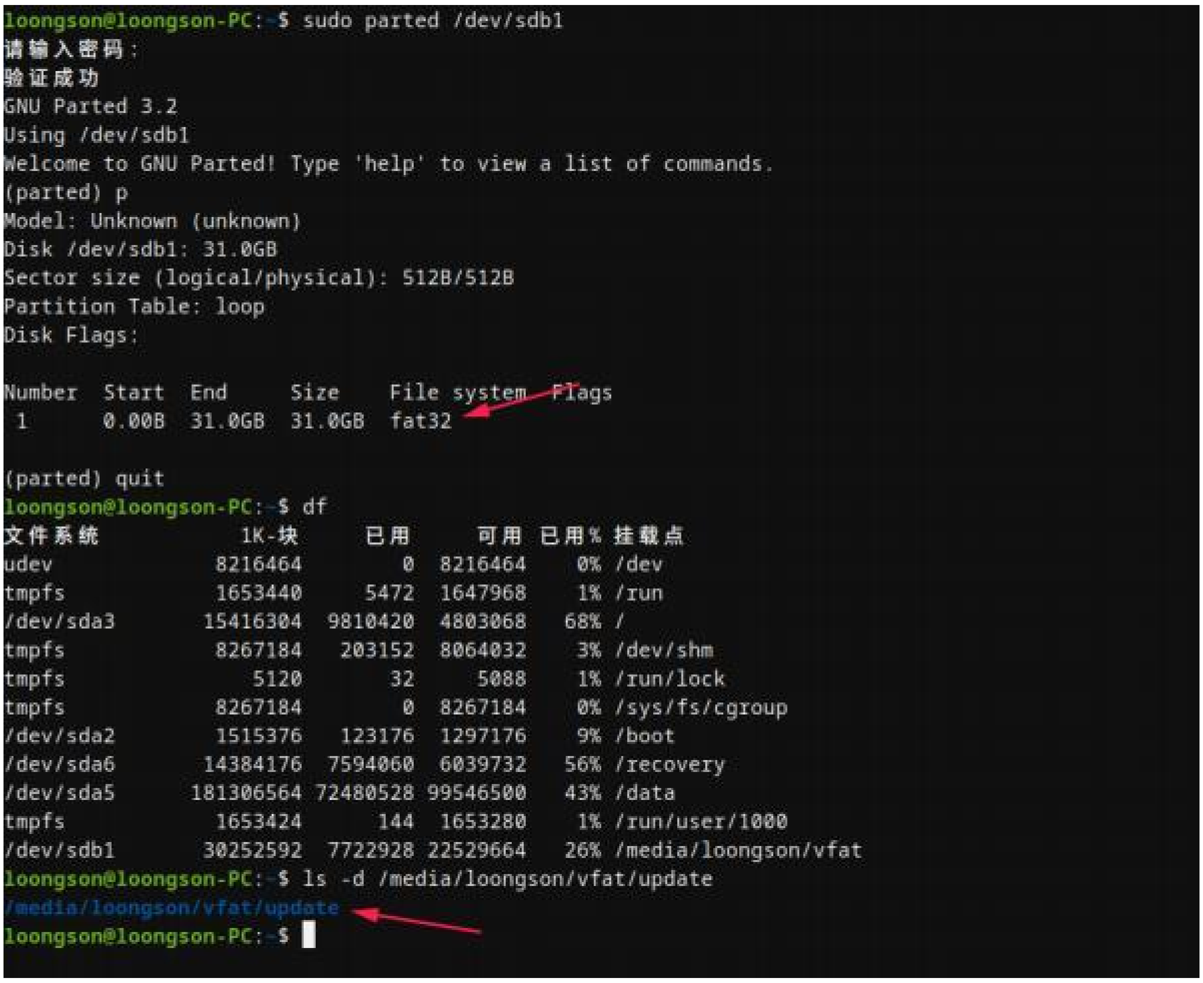
设备 分区 1 / 8.1 GB 驱动器 — Netac OnlyDisk

卷 test20

已用 6.1 GB (75.6%)

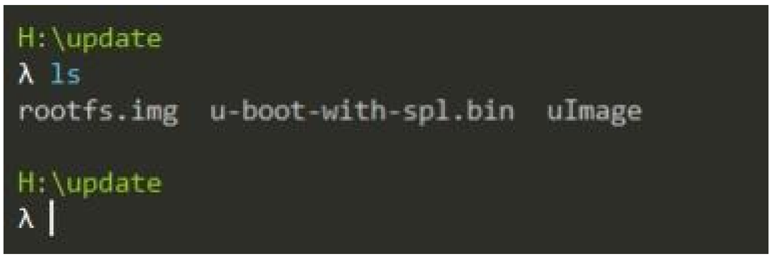
位置 /dev/sdb1

2.在U盘目录下创建update文件夹



```
loongson@loongson-PC:~$ sudo parted /dev/sdb1
请输入密码：
验证成功
GNU Parted 3.2
Using /dev/sdb1
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Model: Unknown (unknown)
Disk /dev/sdb1: 31.0GB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:

Number  Start  End     Size    File system  Flags
 1      0.00B  31.0GB  31.0GB  fat32

(parted) quit
loongson@loongson-PC:~$ df
文件系统            1K-块      已用      可用 已用% 挂载点
udev              8216464         0   8216464    0% /dev
tmpfs             1653440      5472   1647968    1% /run
/dev/sda3        15416304   9810420   4803068   68% /
tmpfs             8267184    203152   8064032    3% /dev/shm
tmpfs                5120        32      5088    1% /run/lock
tmpfs             8267184         0   8267184    0% /sys/fs/cgroup
/dev/sda2         1515376    123176   1297176    9% /boot
/dev/sda6        14384176   7594060   6039732   56% /recovery
/dev/sda5       181306564  72480528  99546500   43% /data
tmpfs             1653424       144   1653280    1% /run/user/1000
/dev/sdb1        30252592   7722928  22529664   26% /media/loongson/vfat
loongson@loongson-PC:~$ ls -d /media/loongson/vfat/update
/media/loongson/vfat/update
loongson@loongson-PC:~$ 
```

3.在U盘update目录下放入要更新的内核（文件名为： uImage）、固件（文件名为： u-boot-with- spl.bin 或 u-boot.bin）、文件系统(rootfs.img)。



```
H:\update
λ ls
rootfs.img  u-boot-with-spl.bin  uImage

H:\update
λ 
```

3.将U盘插在开发板上

4.3.1.2.uboot下尝试读写U盘

1.重新扫描USB设备

```
=> usb reset
resetting USB...

Bus ehci@0x16080000: USB EHCI

1.00 Bus ohci@0x16088000: USB

OHCI 1.0

Bus otg@0x16040000: dwc2_usb otg@0x16040000: Core Release: 2.93a

USB DWC2

scanning  bus  ehci@0x16080000  for  devices...  1  USB  Device(s)  found

scanning  bus  ohci@0x16088000  for  devices...  1  USB  Device(s)  found

scanning bus otg@0x16040000 for devices... 2 USB Device(s) found

        scanning usb for storage devices... 1 Storage Device(s) found
```

2.我们已经在U盘上放置了一些文件（u-boot-with-spl.bin 、uImage、 rootfs.img），尝试列出文件：

```
=> fatls usb 0:1 /update
                  ./
                  ../
  52168418   rootfs.img
   8830630   uImage
    906162   u-boot-with-spl.bin

3 file(s), 2 dir(s)
```

3.尝试读取一个文件到内存：

```
=> fatload usb 0:1 ${loadaddr} /update/uImage
8830630 bytes read in 206 ms (40.9 MiB/s)
=> printenv loadaddr
loadaddr=0x9000000003000000
```

4.这样 U 盘中的文件就被写入了 loadaddr 这一块地址

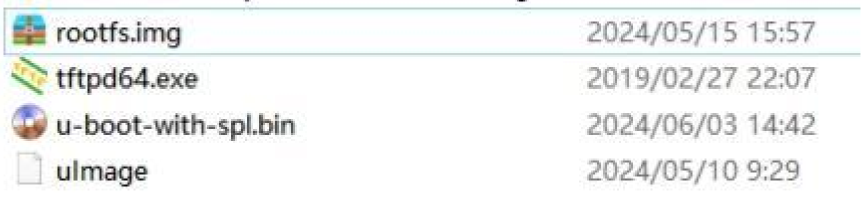
> 若出现U盘无法读写等情况，建议检查U盘分区状态，建议用MBR分区表、单分区格式

4.3.2.tftp 服务

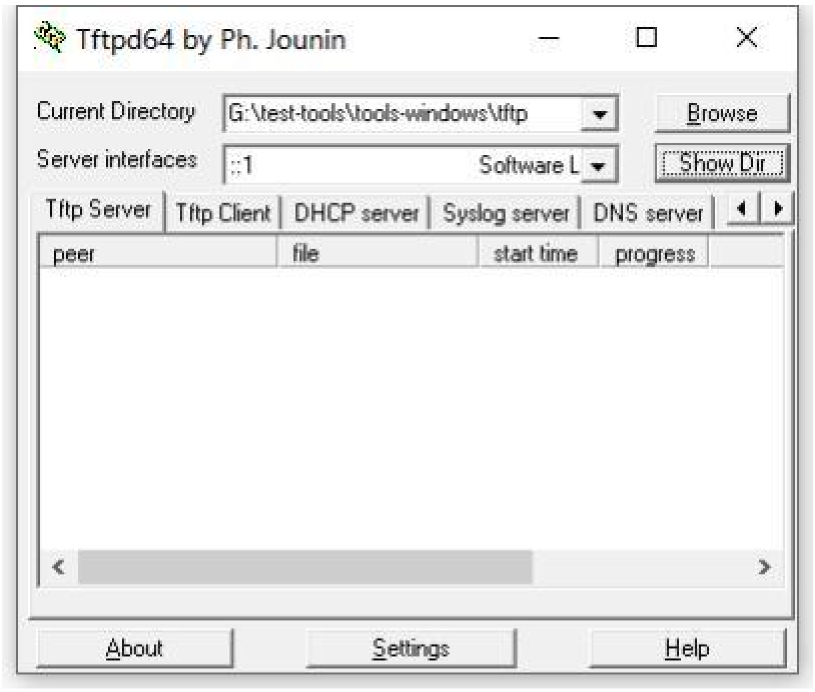
4.3.2.1.安装tftp服务

a.windows平台tftp服务器

windows平台使用tftpd.exe(32位或64位版本都可以，此处用的是64位的版本)，**要更新的文件(内核（文 件名为： uImage）、固件（文件名为： u-boot-with-spl.bin）、文件系统(rootfs.img))和tftpd.exe** **放在同一目录下**，然后打开tftp.exe 即可。



```
rootfs.img            2024/05/15 15:57
tftpd64.exe           2019/02/27 22:07
u-boot-with-spl.bin   2024/06/03 14:42
uImage                2024/05/10 9:29
```



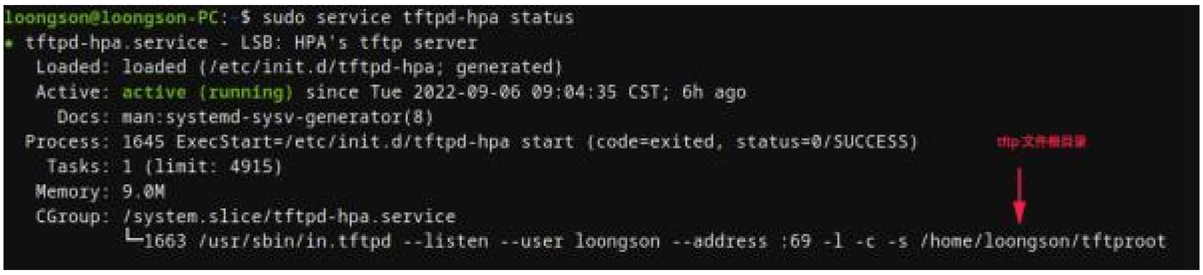
Tftpd64 by Ph. Jounin

Current Directory: G:\test-tools\tools-windows\tftp

Server interfaces: ::1 Software L

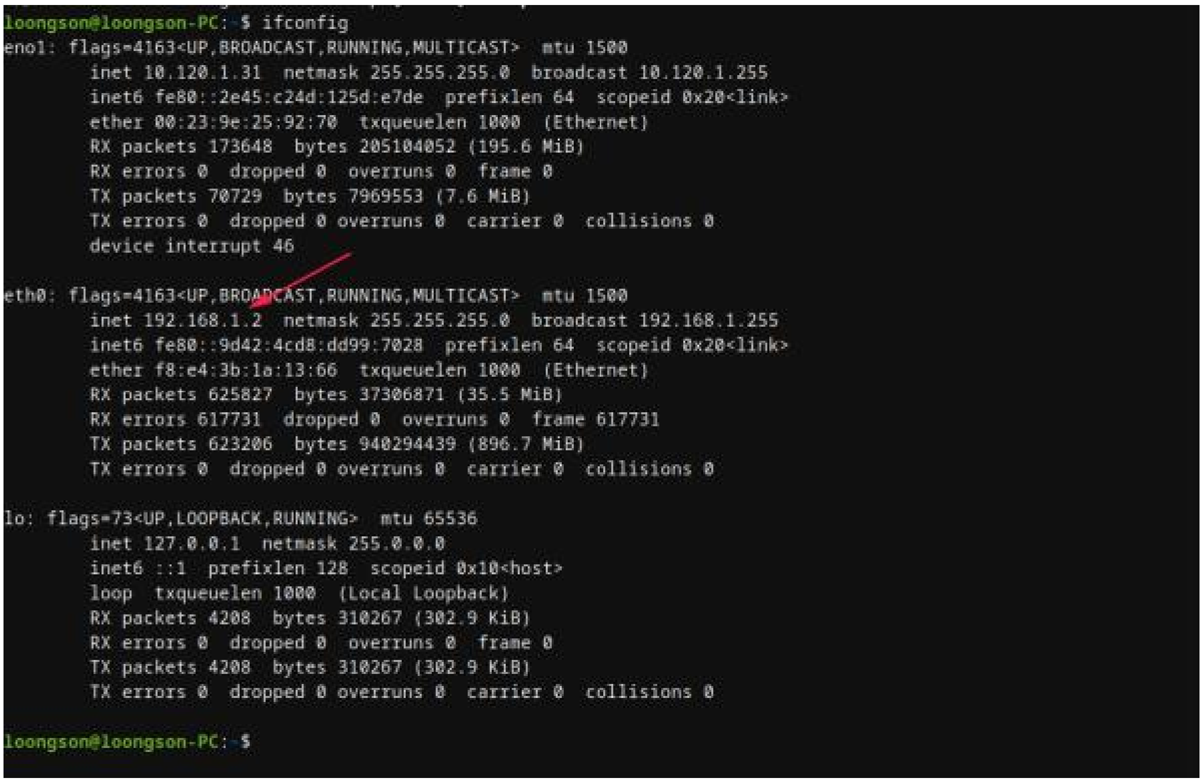
b.linux平台TFTP服务器

1.linux平台TFTP服务器（示例为tftpd-hpa服务），此处设置的tftp文件根目录为`/home/loongson/tftproot/`，**要更新的文件(内核（文件名为： uImage）、固件（文件名为： u-boot-with-spl.bin 或 u-boot.bin）、文件系统(rootfs.img))放在此目录下即可**



```
loongson@loongson-PC:~$ sudo service tftpd-hpa status
● tftpd-hpa.service - LSB: HPA's tftp server
   Loaded: loaded (/etc/init.d/tftpd-hpa; generated)
   Active: active (running) since Tue 2022-09-06 09:04:35 CST; 6h ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1645 ExecStart=/etc/init.d/tftpd-hpa start (code=exited, status=0/SUCCESS)
    Tasks: 1 (limit: 4915)
   Memory: 9.0M
   CGroup: /system.slice/tftpd-hpa.service
           └─1663 /usr/sbin/in.tftpd --listen --user loongson --address :69 -l -c -s /home/loongson/tftproot
```

2.查看服务器IP地址，比如 192.168.1.2



```
loongson@loongson-PC:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.120.1.31  netmask 255.255.255.0  broadcast 10.120.1.255
        inet6 fe80::2e45:c24d:125d:e7de  prefixlen 64  scopeid 0x20<link>
        ether 00:23:9e:25:92:70  txqueuelen 1000  (Ethernet)
        RX packets 173648  bytes 205104052 (195.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 70729  bytes 7969553 (7.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 46

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.2  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::9d42:4cd8:dd99:7028  prefixlen 64  scopeid 0x20<link>
        ether f8:e4:3b:1a:13:66  txqueuelen 1000  (Ethernet)
        RX packets 625827  bytes 37306871 (35.5 MiB)
        RX errors 617731  dropped 0  overruns 0  frame 617731
        TX packets 623206  bytes 940294439 (896.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4208  bytes 310267 (302.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4208  bytes 310267 (302.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

loongson@loongson-PC:~$
```

3.网线连接服务器与开发板，准备更新或传输文件

4.3.2.2.uboot网络设置（可选）

1.查看"tftp服务器ip"与"本机ip"

"tftp服务器ip"与"本机ip"都存储在环境变量中，其中，变量 **serverip** 是"tftp服务器ip"，变量 **ipaddr** 是 "本机ip"，可在 `uboot console` 下执行 `printenv` 查看环境变量。

```
=> printenv

...

ipaddr=192.168.1.20

...

serverip=192.168.1.2

...


Environment size: 1315/16380 bytes

=>
```

2.更改服务器ip与本机ip

```
=> setenv serverip 192.168.2.2 =>
```

3.更改本机ip

```
=> setenv ipaddr 192.168.2.20

=>
```

4.网关配置

如果服务器与设备ip处在不同网段，则需设置网关ip。网关ip对应变量 **gatewayip**。

```
=> setenv gatewayip 192.168.1.1

=>
```

5.保存更改后的配置

ip更改后，可以执行 **saveenv** 长期保存

```
=> saveenv
Saving Environment to SPIFlash... Erasing SPI flash...Writing to SPI flash...done OK

=>
```

4.3.2.3.uboot下尝试tftp

我们已经在tftp服务器上放置了一些文件（u-boot-with-spl.bin 、uImage、 rootfs.img ），尝试以下操 作：

```
=> tftp uImage

Speed: 1000, full duplex

Using ethernet@0x16020000 device

TFTP from server 192.168.1.2; our IP address is 192.168.1.20

Filename 'uImage'.

Load address: 0x9000000003000000
Loading:
####################################################################
###

            ...

            ...

             #####################################

             4.9 MiB/s

done

Bytes transferred = 8826725 (86af65 hex)

=>
```

注：

若出现网络下载或更新不了的情况，建议检查电脑的防火墙是否为打开状态，如果是打开状态则将其关闭之后再试。

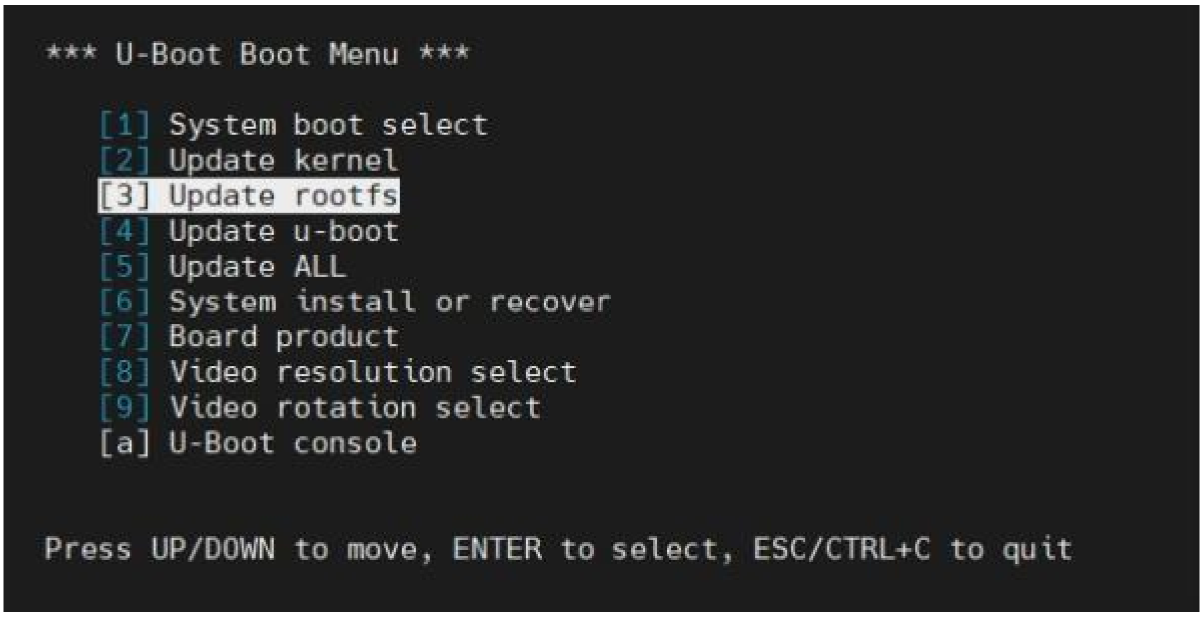
4.4.EMMC使用方法

将安装系统到EMMC提供有两种方法，当因系统镜像过大（大于 152MB）导致安装失败时可使用方法 2。

4.4.1.EMMC安装系统方法1

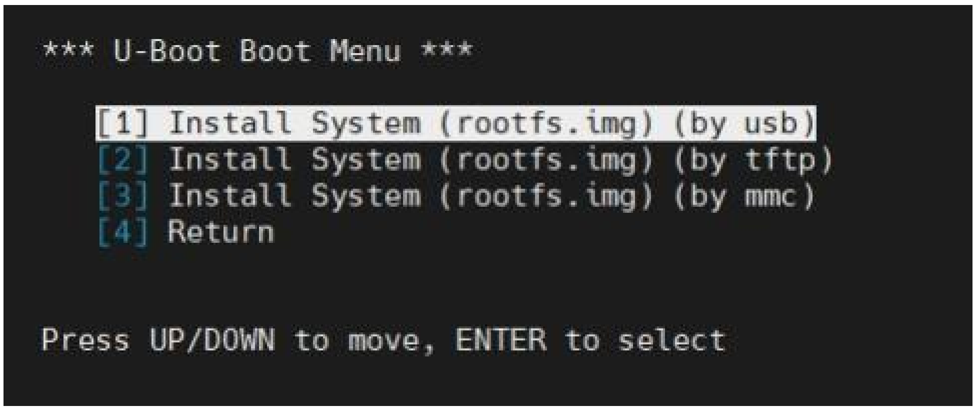
1.将文件系统 **文件系统(rootfs.img)** 放在tftp服务器**根目录**下，或者U盘**update**目录下。板卡插上网线 或U盘

2.uboot菜单选择“[3] Update rootfs”

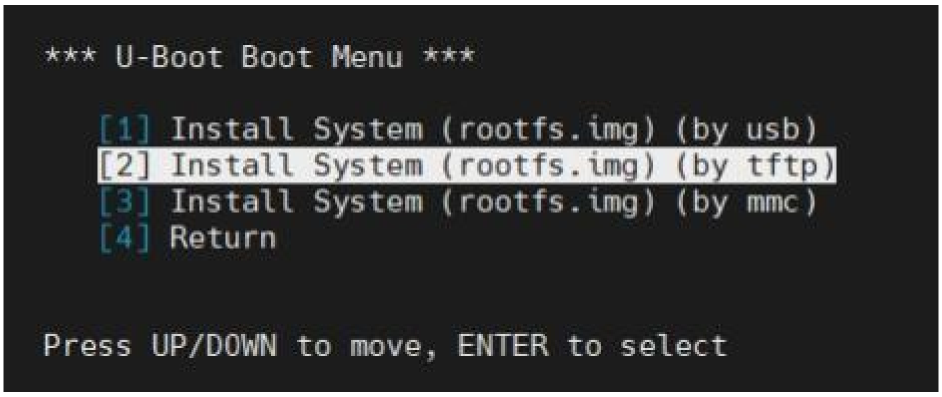


```
*** U-Boot Boot Menu ***

    [1] System boot select
    [2] Update kernel
    [3] Update rootfs
    [4] Update u-boot
    [5] Update ALL
    [6] System install or recover
    [7] Board product
    [8] Video resolution select
    [9] Video rotation select
    [a] U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

3.选择“[1] Update rootfs (rootfs.img) (by usb)”或“[2] Update rootfs (rootfs.img) (by tftp)”，开始升 级：

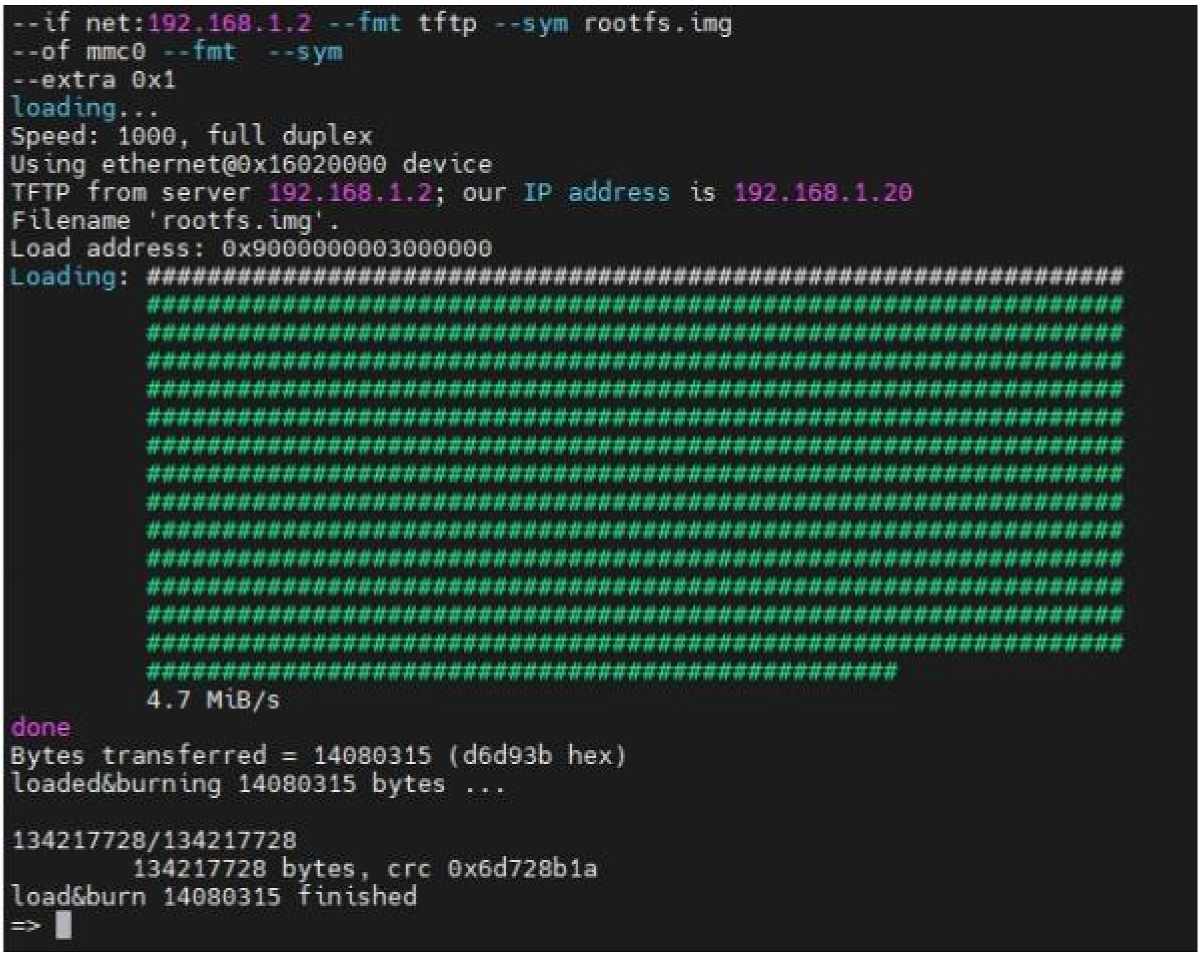


```
*** U-Boot Boot Menu ***

    [1] Install System (rootfs.img) (by usb)
    [2] Install System (rootfs.img) (by tftp)
    [3] Install System (rootfs.img) (by mmc)
    [4] Return


Press UP/DOWN to move, ENTER to select
```



```
*** U-Boot Boot Menu ***

    [1] Install System (rootfs.img) (by usb)
    [2] Install System (rootfs.img) (by tftp)
    [3] Install System (rootfs.img) (by mmc)
    [4] Return


Press UP/DOWN to move, ENTER to select
```

4.升级完成，重启输入 `reboot`或 `reset`



```
--if net:192.168.1.2 --fmt tftp --sym rootfs.img
--of mmc0 --fmt  --sym
--extra 0x1
loading...
Speed: 1000, full duplex
Using ethernet@0x16020000 device
TFTP from server 192.168.1.2; our IP address is 192.168.1.20
Filename 'rootfs.img'.
Load address: 0x9000000003000000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #####################################################
         4.7 MiB/s
done
Bytes transferred = 14080315 (d6d93b hex)
loaded&burning 14080315 bytes ...

134217728/134217728
        134217728 bytes, crc 0x6d728b1a
load&burn 14080315 finished
=>
```

4.4.2.EMMC安装系统方法2

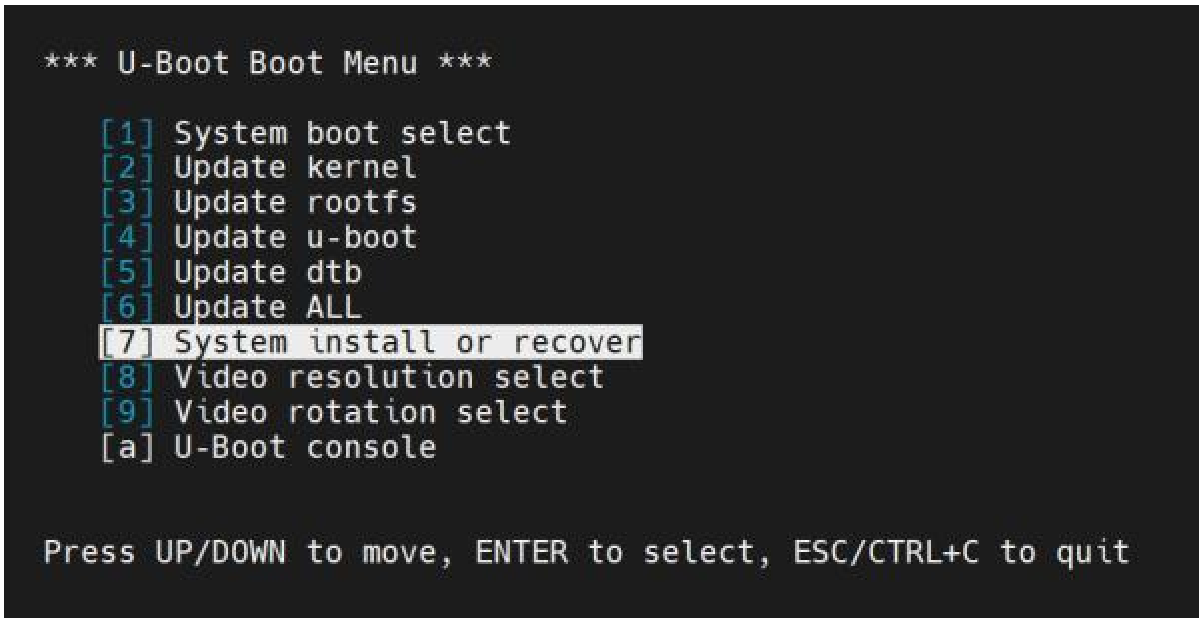
4.4.2.1.利用rootfs.tar.gz向EMMC中安装系统步骤

1.将以下三个文件 **内核（uImage）**、**文件系统压缩包(rootfs.tar.gz)** 以及 **ramdisk.gz** 放在tftp服务器 **根目录**下，或者U盘 **install** 目录下。

注:

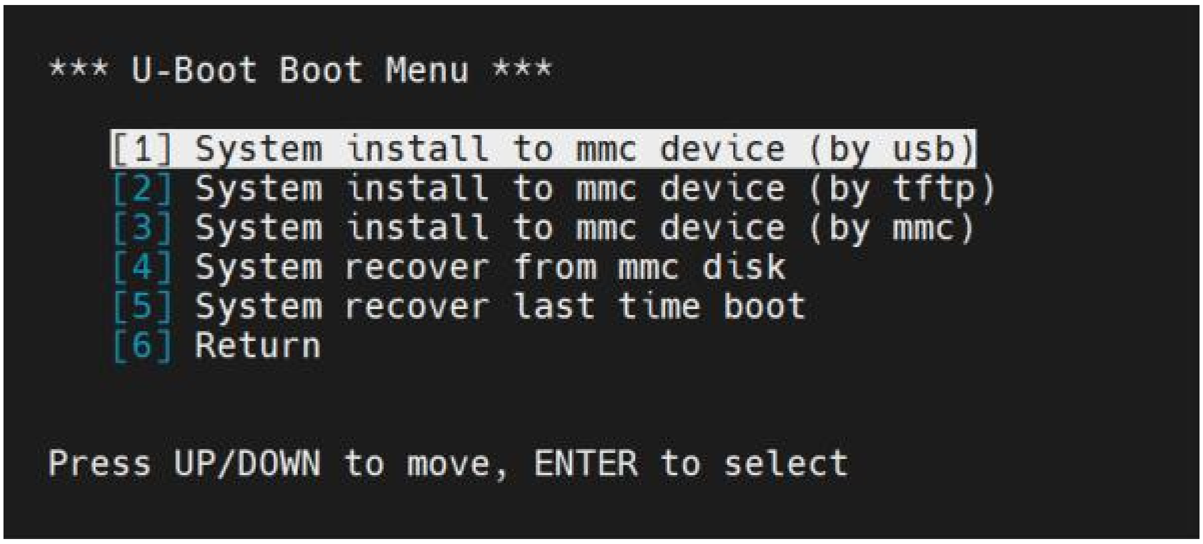
> 放在U盘 *install* 目录下，不再是 *update* 目录。

2.uboot菜单选择“[6] System install or recover”

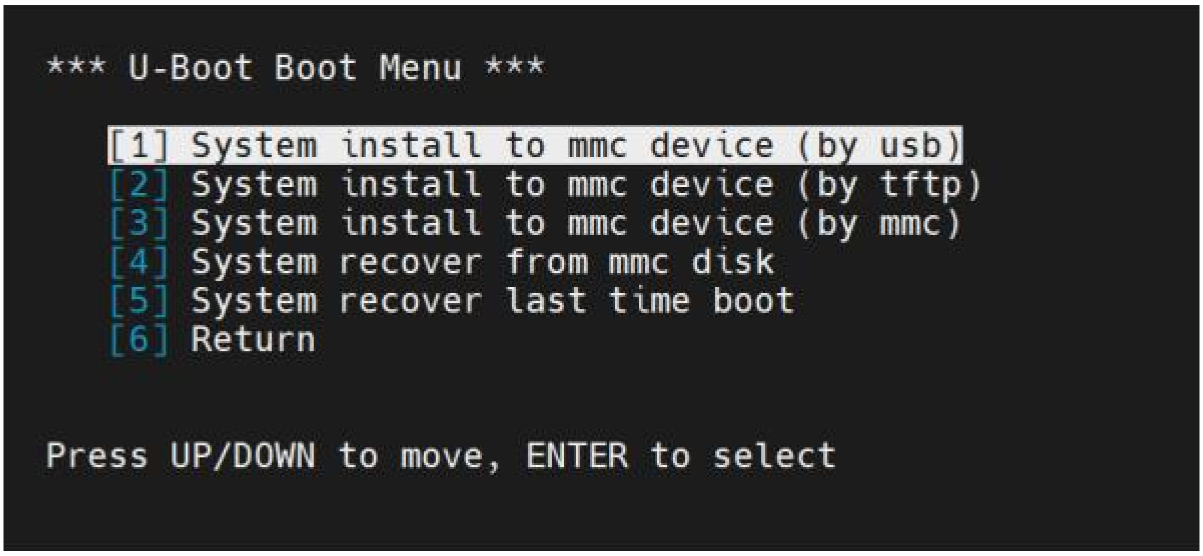


```
*** U-Boot Boot Menu ***

   [1] System boot select
   [2] Update kernel
   [3] Update rootfs
   [4] Update u-boot
   [5] Update dtb
   [6] Update ALL
   [7] System install or recover
   [8] Video resolution select
   [9] Video rotation select
   [a] U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

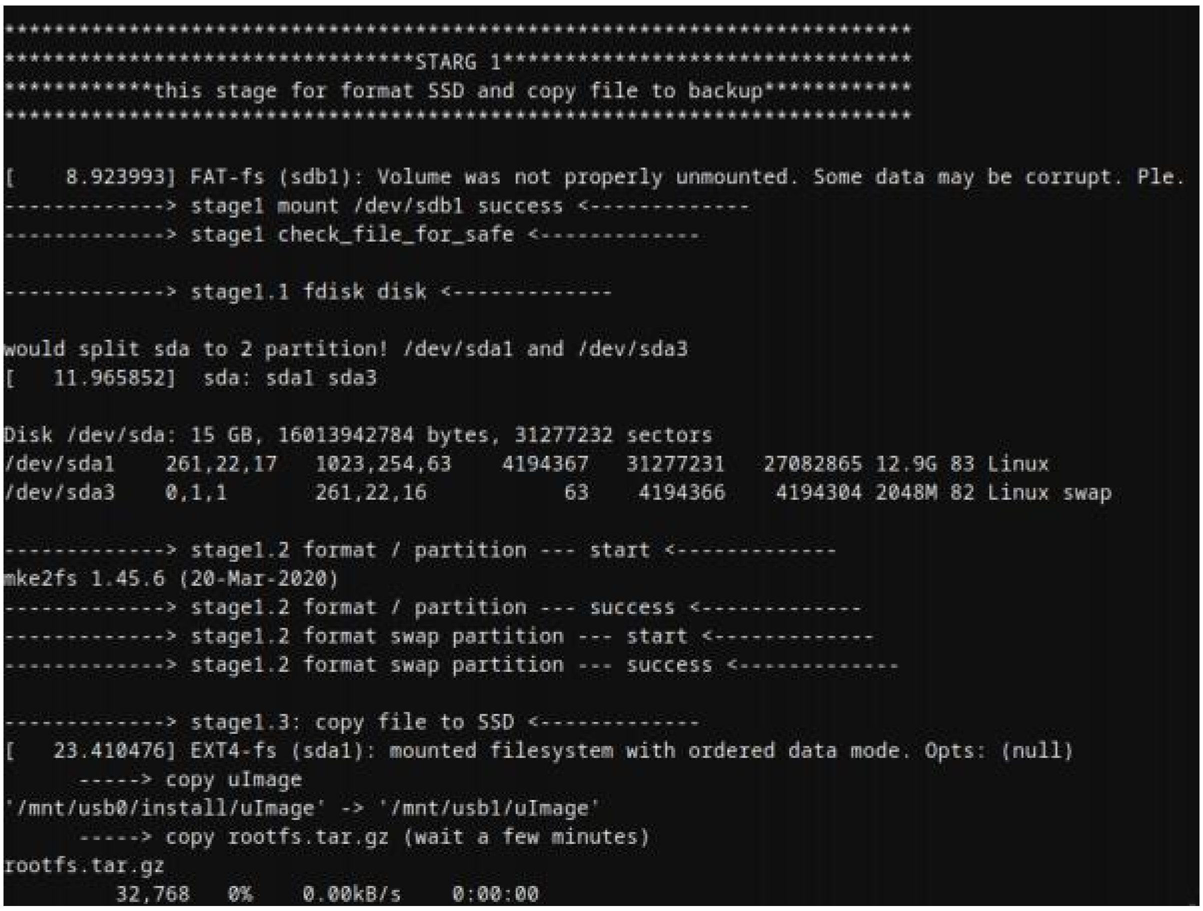
3.选择“[1] System install to mmc (by usb)”或“[3] Update u-boot to mmc (by tftp)”，开始安装：



```
*** U-Boot Boot Menu ***

    [1] System install to mmc device (by usb)
    [2] System install to mmc device (by tftp)
    [3] System install to mmc device (by mmc)
    [4] System recover from mmc disk
    [5] System recover last time boot
    [6] Return


Press UP/DOWN to move, ENTER to select
```



```
*** U-Boot Boot Menu ***

    [1] System install to mmc device (by usb)
    [2] System install to mmc device (by tftp)
    [3] System install to mmc device (by mmc)
    [4] System recover from mmc disk
    [5] System recover last time boot
    [6] Return


Press UP/DOWN to move, ENTER to select
```



```
************************************************************************
*********************************STARG 1********************************
************this stage for format SSD and copy file to backup************
************************************************************************

[    8.923993] FAT-fs (sdb1): Volume was not properly unmounted. Some data may be corrupt. Ple.
-------------> stage1 mount /dev/sdb1 success <-------------
-------------> stage1 check_file_for_safe <-------------

-------------> stage1.1 fdisk disk <-------------

would split sda to 2 partition! /dev/sda1 and /dev/sda3
[   11.965852]  sda: sda1 sda3

Disk /dev/sda: 15 GB, 16013942784 bytes, 31277232 sectors
/dev/sda1    261,22,17   1023,254,63    4194367   31277231   27082865 12.9G 83 Linux
/dev/sda3    0,1,1       261,22,16           63    4194366    4194304 2048M 82 Linux swap

-------------> stage1.2 format / partition --- start <-------------
mke2fs 1.45.6 (20-Mar-2020)
-------------> stage1.2 format / partition --- success <-------------
-------------> stage1.2 format swap partition --- start <-------------
-------------> stage1.2 format swap partition --- success <-------------

-------------> stage1.3: copy file to SSD <-------------
[   23.410476] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null)
      -----> copy uImage
'/mnt/usb0/install/uImage' -> '/mnt/usb1/uImage'
      -----> copy rootfs.tar.gz (wait a few minutes)
rootfs.tar.gz
         32,768   0%    0.00kB/s    0:00:00
```

安装成功之后系统会自动重启。

4.4.2.2.rootfs.tar.gz分区布局说明

rootfs.tar.gz中不包含分区信息，默认一个分区，下面说明具体的分区控制情况：

表5-3 开发板的分区用途说明表

分区名	用途
/dev/mmcblk0p1	文件系统的主分区，即 / 所挂载的分。这是**必要**的分区。
/dev/mmcblk0p2	文件系统的数据分区，即把/home /opt /var 分区的内容放置到这个分区中， 实现系统文件和数据文件分离的效果。这是**非必要**的分区。
/dev/mmcblk0p3	交换分区，默认大小为2G。这是**非必要**的分区。
/dev/mmcblk0p4	备 份 分 区 ， 存 放 rootfs. tar. gz、 ramdisk. gz、 uImage。 uboot中可 以 选 择 恢 复文件系统，则使用以上文件进行文件系统的恢复，恢复的是文件系统的主 分区(/dev/mmcblk0p1)。 上述中的ramdisk.gz是引导系统，类似于WinPE。用于安装，恢复文件系统 的系统。 同时，如果选择双系统策略，通过龙芯软件则可以让此分区变成一个文件系统的主分区。这是**非必要**的分区。

上述的表格中可见分区的必要性和分区的作用，也有很多分区策略。对此龙芯的引导系统支持默认的分区策略和自定义分区策略来进行分区。

下面将说明**默认的分区策略**的代号及其含义。请关注这点，这对于安装系统的时候有帮助。

表5-4 默认分区策略的代号和含义对照表

代号	含义
无代号 默认情况	只划分/dev/mmcblk0p1和/dev/mmcblk0p3。 /dev/mmcblk0p3默认为2GB（用做交换分区）。 然后把剩余所有的空间划分给/dev/mmcblk0p1。
4part	划分/dev/mmcblk0p1、/dev/mmcblk0p2、/dev/mmcblk0p3 和/dev/mmcblk0p4。 /dev/mmcblk0p3默认为2GB（用做交换分区）。 /dev/mmcblk0p4默认为4G。 剩余的空间按照1 ：2的比例分到/dev/mmcblk0p1 和 /dev/mmcblk0p2中
twosys	划分/dev/mmcblk0p1、/dev/mmcblk0p2、/dev/mmcblk0p3 和/dev/mmcblk0p4。 /dev/mmcblk0p3默认为2GB（用做交换分区）。 剩余空间按照1 ：2 ：3的比例分到/dev/mmcblk0p1、/dev/mmcblk0p2和/dev/mmcblk0p4中。 这个策略是为了/dev/mmcblk0p4可以用作主分区。

代号	含义
twosys_3	划分/dev/mmcblk0p1、/dev/mmcblk0p3、/dev/mmcblk0p4。 /dev/mmcblk0p3默认为2GB（用做交换分区）。 剩余空间按照1 ： 1的比例分到 /dev/mmcblk0p1 和 /dev/mmcblk0p4中。 这个策略是为了/dev/mmcblk0p4可以用作主分区。

下面将说明**自定义分区策略**的含义，但相信上述的默认分区策略已经足够满足大多数场景。

自定义分区策略支持使用数字来描述对于分区的大小比例。详细说明如下：

表5-5 分区大小比例及其选择范围对照表

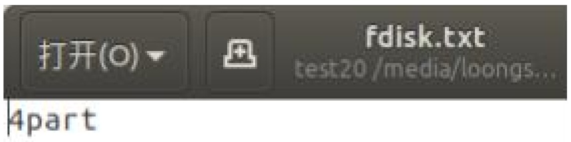
分区名	含义
/dev/mmcblk0p1	该数字一定要大于 0
/dev/mmcblk0p2	可以选择大于等于 0 的数字
/dev/mmcblk0p3	-1 代表分 2G 0 代表不创建 不能大于 0
/dev/mmcblk0p4	-2 代表分 5G -1 代表分 4G 0 代表不创建 可以大于 0

也就是使用四个数字来说明分区的比例。大于0的数字代表比例。 0代表不创建，小于0的数字代表默认的 大小。

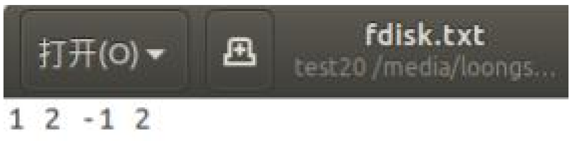
比如1 2 -1 2。就代表/dev/mmcblk0p1、/dev/mmcblk0p2、/dev/mmcblk0p4按照1 ： 2 ： 2的比例划 分。/dev/mmcblk0p3为2G。

上述说到的**代号**和**比例划分**，需要提前准备好在USB或者是tftp服务端文件夹下。方法如下：

创建一个fdisk.txt文件，里面只填写一行内容，那一行内容可以是**代号**，也可以是**比例描述**。例子见下 图：



```
4part
```



```
1 2 -1 2
```

对于使用U盘方式安装，可以直接创建一个名字为**代号**的文件夹，例子见下图：

注意不能创建多个不同代号的文件夹，否则不能按照预期的想法分区。也需要**注意fdisk.txt文件的优先 级大于指定文件夹的优先级**。

4.4.3. EMMC更新内核（最简单操作）

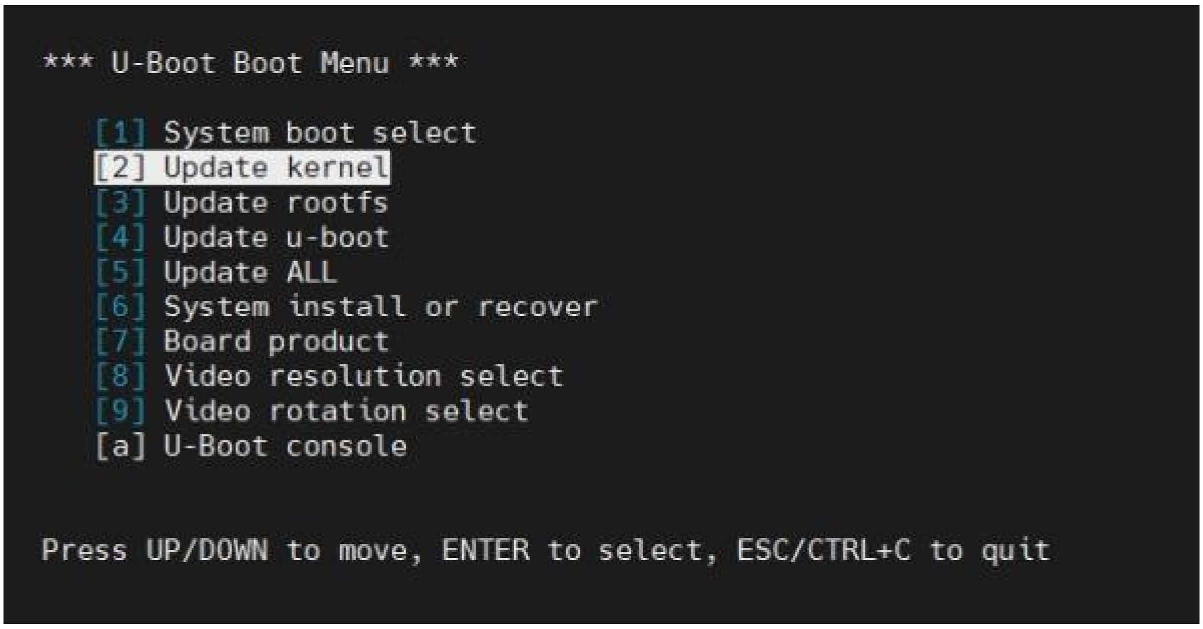
1、从EMMC启动，进入系统

2、直接将 **内核（uImage）** 替换到 **/boot** 下，重启

4.4.4. EMMC通过uboot更新内核

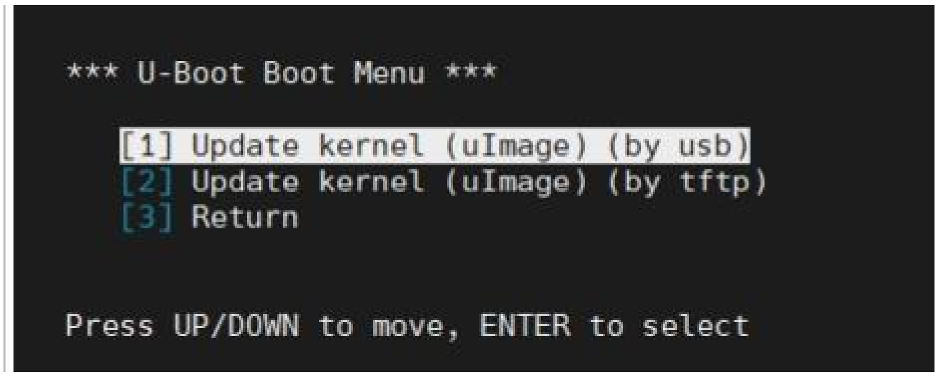
1.将 **内核（uImage）** 放在tftp服务器**根目录**下，或者U盘**update**目

录下。 2.uboot菜单选择“[2] Update kernel”

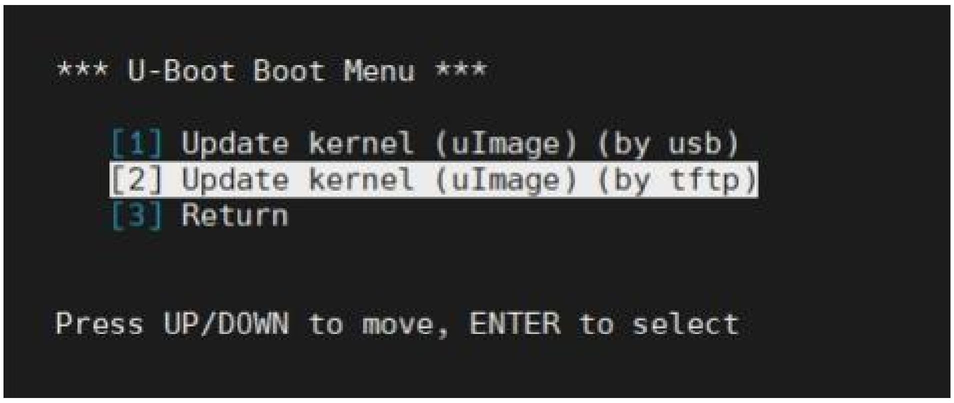


```
*** U-Boot Boot Menu ***

    [1] System boot select
    [2] Update kernel
    [3] Update rootfs
    [4] Update u-boot
    [5] Update ALL
    [6] System install or recover
    [7] Board product
    [8] Video resolution select
    [9] Video rotation select
    [a] U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

3.选择“[1] Update kernel (uImage) (by usb)”或“[2] Update kernel (uImage) (by tftp)”，开始升级：

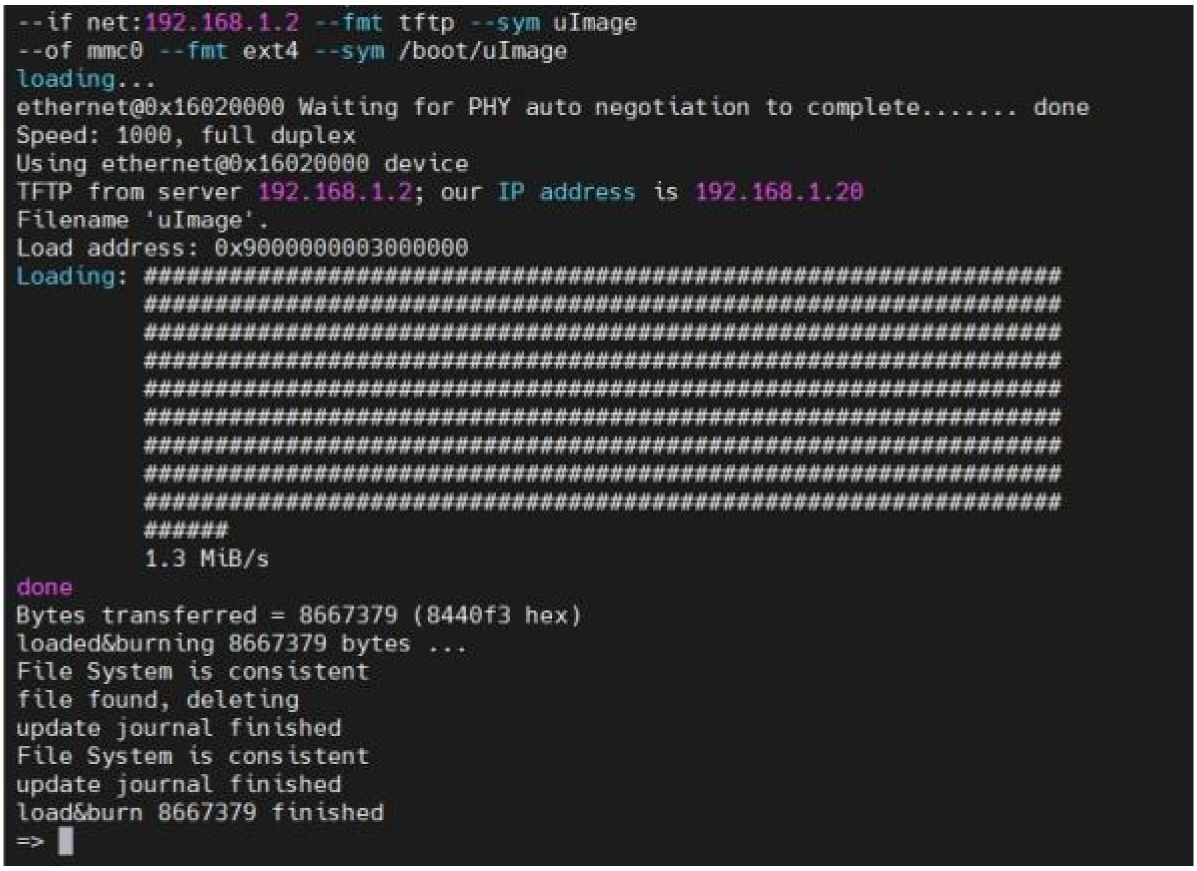


```
*** U-Boot Boot Menu ***

    [1] Update kernel (uImage) (by usb)
    [2] Update kernel (uImage) (by tftp)
    [3] Return


Press UP/DOWN to move, ENTER to select
```



```
*** U-Boot Boot Menu ***

    [1] Update kernel (uImage) (by usb)
    [2] Update kernel (uImage) (by tftp)
    [3] Return


Press UP/DOWN to move, ENTER to select
```

4.升级完成，重启输入 *reboot*或 *reset*



```
--if net:192.168.1.2 --fmt tftp --sym uImage
--of mmc0 --fmt ext4 --sym /boot/uImage
loading...
ethernet@0x16020000 Waiting for PHY auto negotiation to complete....... done
Speed: 1000, full duplex
Using ethernet@0x16020000 device
TFTP from server 192.168.1.2; our IP address is 192.168.1.20
Filename 'uImage'.
Load address: 0x9000000003000000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ######
         1.3 MiB/s
done
Bytes transferred = 8667379 (8440f3 hex)
loaded&burning 8667379 bytes ...
File System is consistent
file found, deleting
update journal finished
File System is consistent
update journal finished
load&burn 8667379 finished
=> 
```

4.5.TF卡使用方法

2K300 蜂鸟板支持从TF卡启动系统， TF卡方便拆下用读卡器连接到PC上进行各种定制修改，且不像 EMMC系统需要关心镜像大小，只需要一个rootfs.img镜像文件即可。

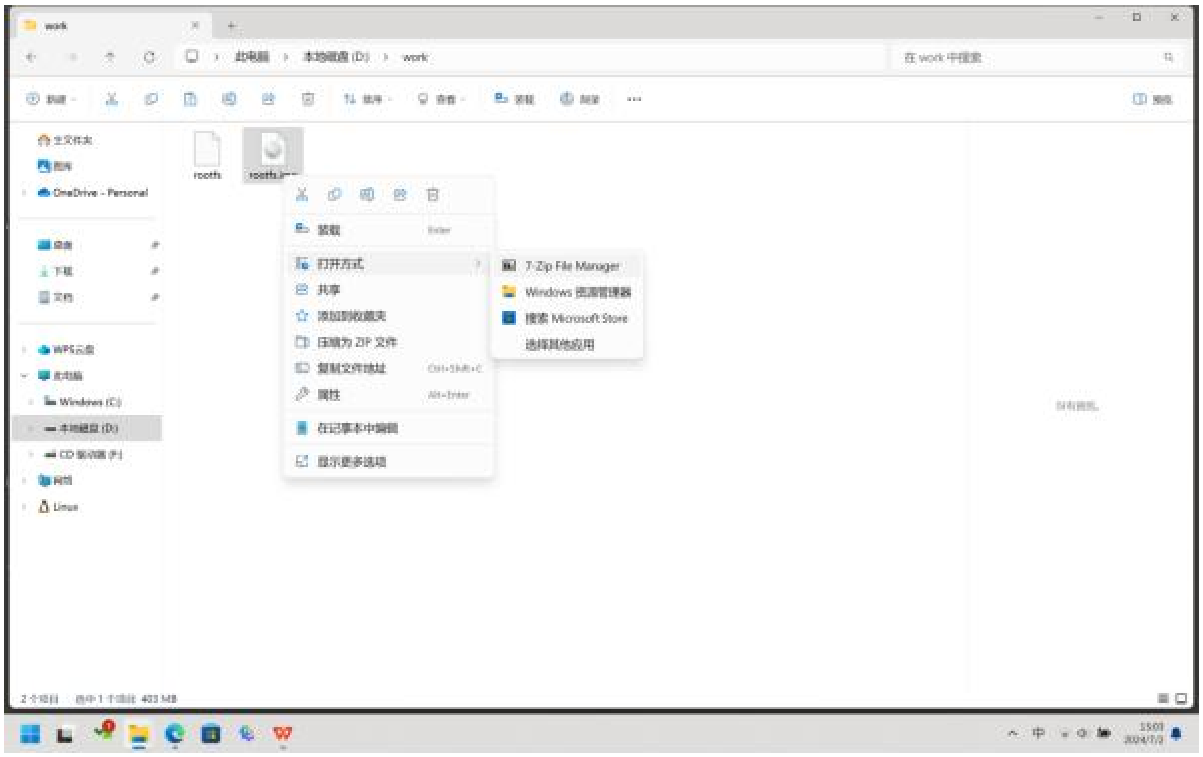
4.5.1. TF卡文件系统安装

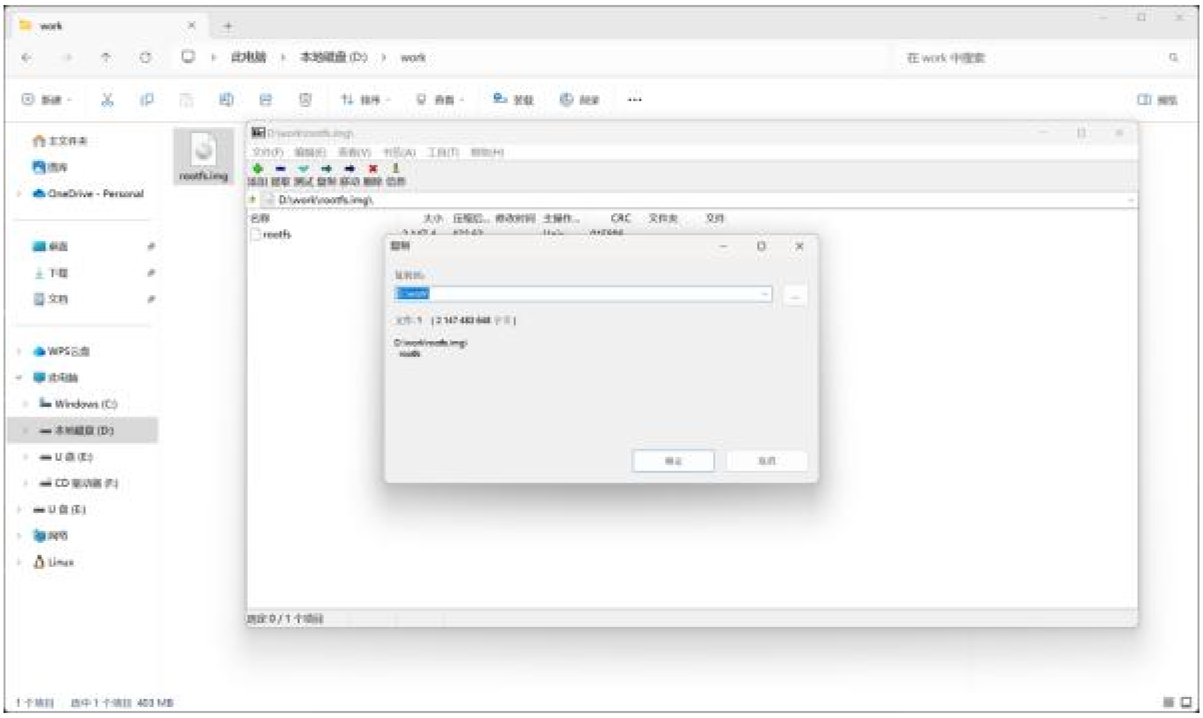
a.Windows下安装TF卡系统

1.提前安装7Z（或其他解压缩工具）、

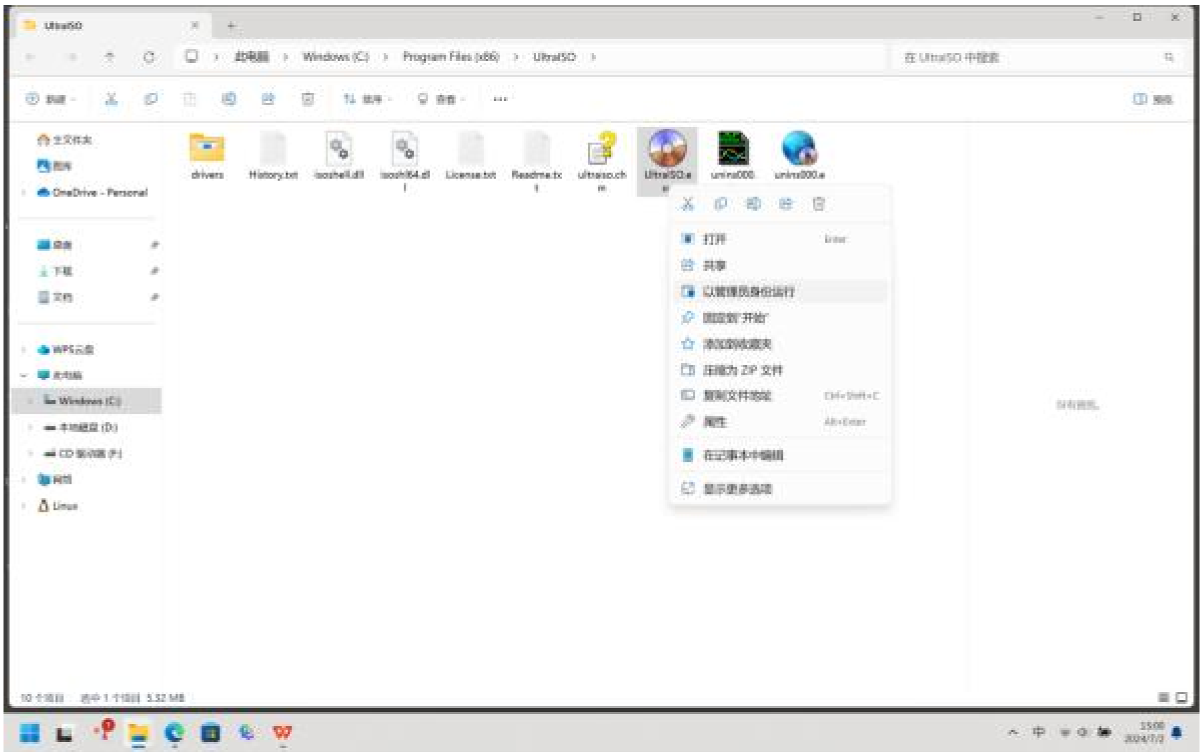
UltraISO工具 2.通过读卡器将TF卡插在PC上

3.用 7Z 或其他解压工具打开 rootfs.img，解压，这里将解压后的文件命名为 rootfs。

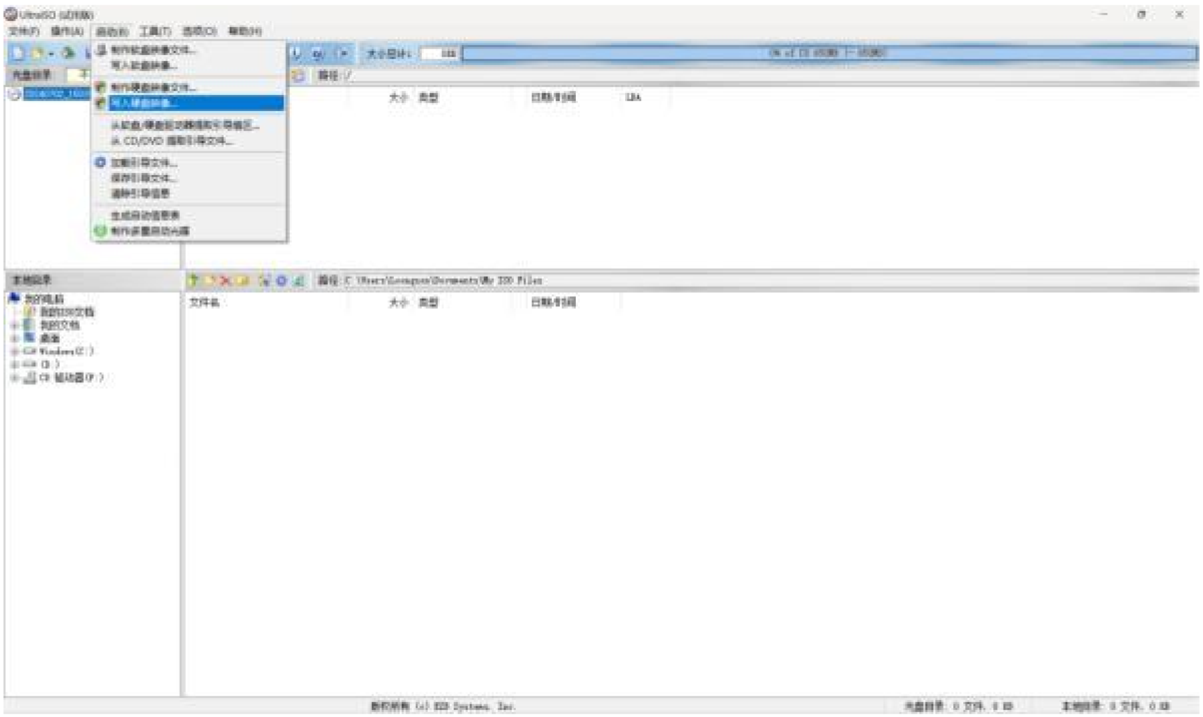




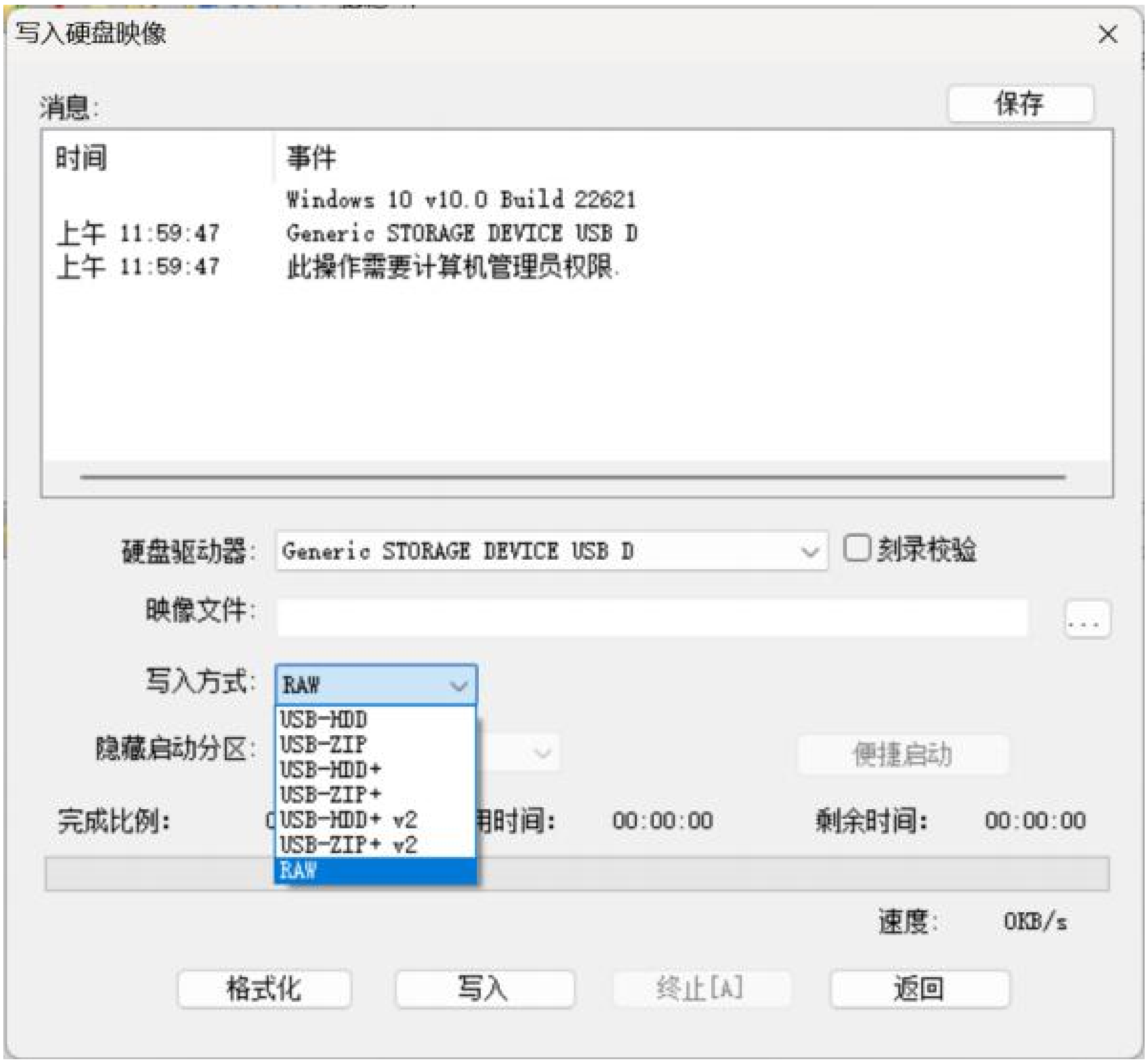
4.以管理员身份运行*ultraiso*工具



5.选择"启动"->"写入硬盘镜像"



6.写入方式改成"RAW"


写入硬盘映像

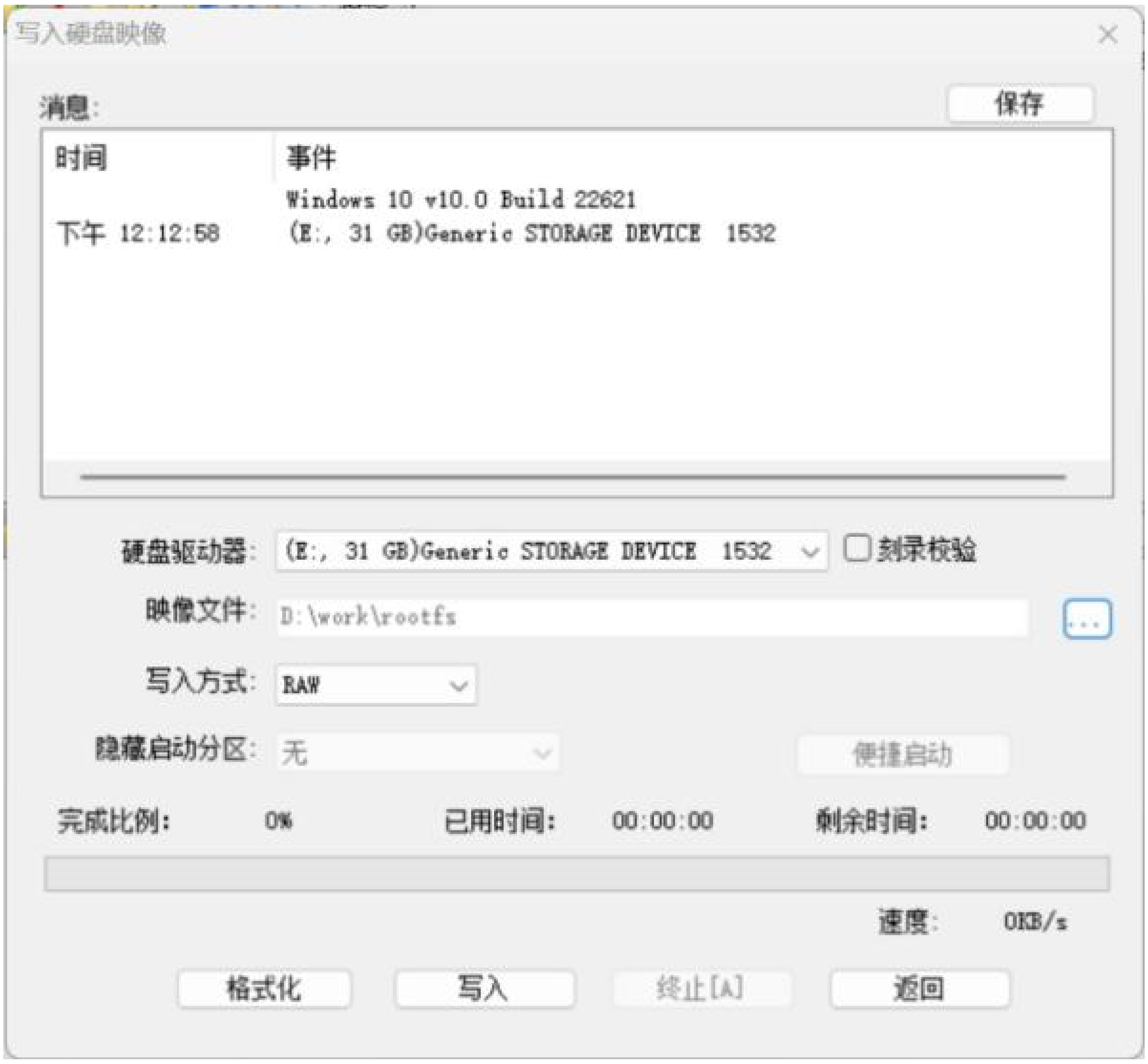
消息：

时间	事件
	Windows 10 v10.0 Build 22621
上午 11:59:47	Generic STORAGE DEVICE USB D
上午 11:59:47	此操作需要计算机管理员权限.

硬盘驱动器：Generic STORAGE DEVICE USB D

写入方式：RAW

7.选择映象文件“*rootfs*”



```
                  Windows 10 v10.0 Build 22621
下午 12:12:58     (E:, 31 GB)Generic STORAGE DEVICE  1532
```

硬盘驱动器: (E:, 31 GB)Generic STORAGE DEVICE 1532 ☐ 刻录校验

映像文件: D:\work\rootfs

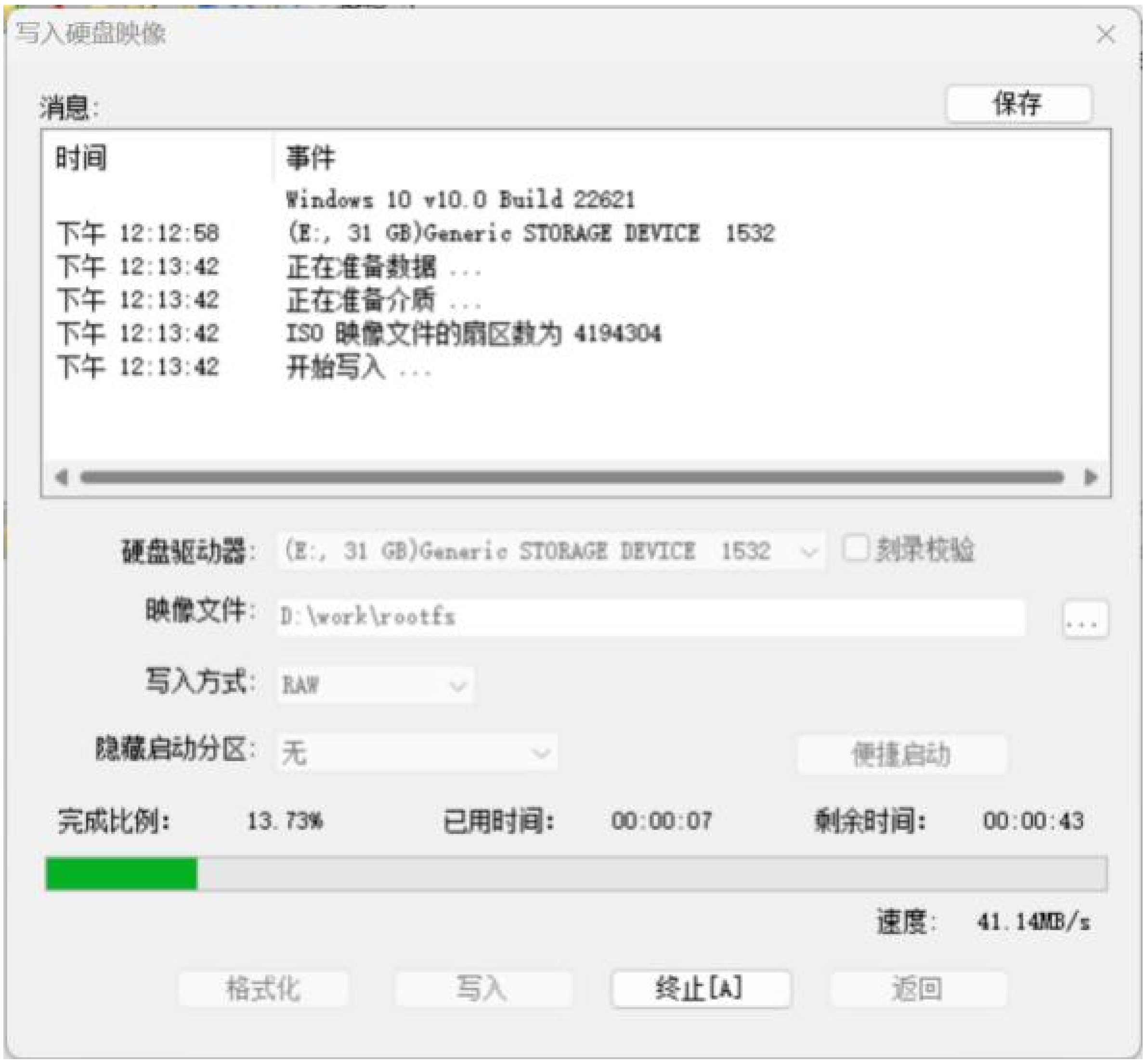
写入方式: RAW

隐藏启动分区: 无

完成比例: 0% 已用时间: 00:00:00 剩余时间: 00:00:00

速度: 0KB/s

8.点击"写入"，等待完成



消息:

```
时间              事件
                  Windows 10 v10.0 Build 22621
下午 12:12:58     (E:, 31 GB)Generic STORAGE DEVICE  1532
下午 12:13:42     正在准备数据 ...
下午 12:13:42     正在准备介质 ...
下午 12:13:42     ISO 映像文件的扇区数为 4194304
下午 12:13:42     开始写入 ...
```

硬盘驱动器: (E:, 31 GB)Generic STORAGE DEVICE 1532

映像文件: D:\work\rootfs

写入方式: RAW

隐藏启动分区: 无

完成比例: 13.73%　已用时间: 00:00:07　剩余时间: 00:00:43

速度: 41.14MB/s

b. Linux下安装TF卡系统

1.通过读卡器将TF卡插在PC上，得到设备符，比如 **/dev/sdX**，并确保没有挂载

```
$ sudo umount /dev/sdX
```

2.接下来进行解压缩与写入操作

```
$ gunzip -S .img rootfs.img
$ sudo dd if=rootfs of=/dev/sdX
```

4.5.2.TF卡更新内核（最简单操作）

1、从TF卡启动，进入系统

2、直接将 **内核（uImage）** 替换到 **/boot** 下，重启

4.5.3.TF卡更新内核

TF卡可以灵活拆下，所以更新TF卡系统内核只需要用读卡器连接到PC上，替换 **boot/uImage** 即可。 这些操作在 linux-PC 上是非常简单的，通过读卡器将TF卡插在PC上，得到设备符，比如 **/dev/sdX**。

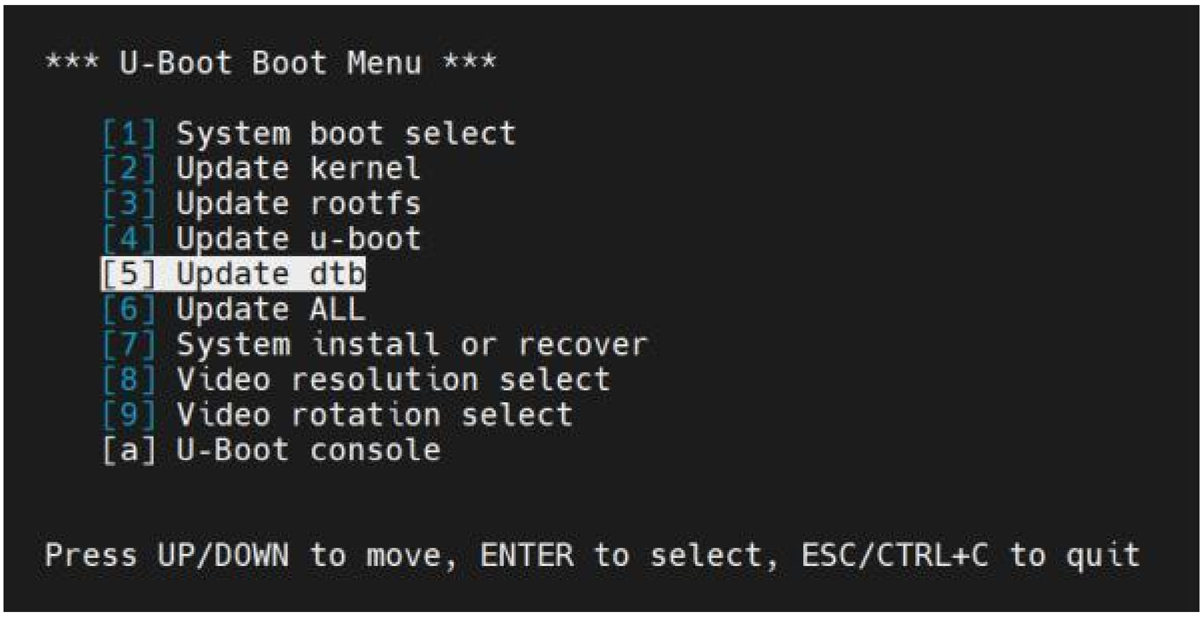
```
$ sudo mount /dev/sdX1 /media
$ sudo cp uImage /media/boot/uImage $
sudo umount /dev/sdX1
```

4.6.设备树（DTB）更新

可以在linux内核中执行 `make dtbs`编译新的dtb，生成需要的 ls2k300_pai.dtb

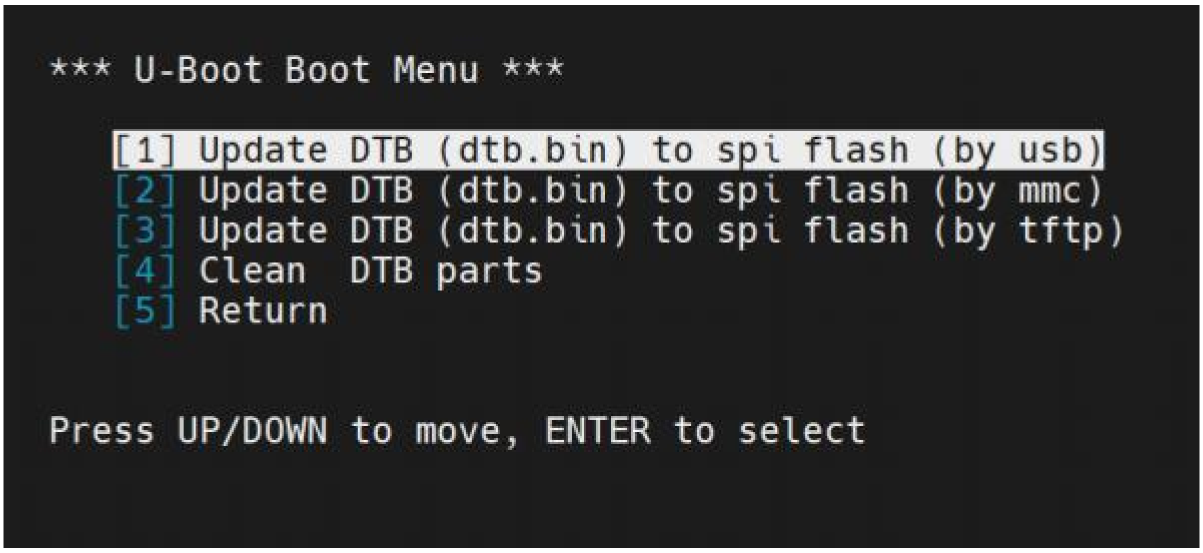
```
make dtbs
```

1.将编译好的 ls2k300_pai.dtb 改名为 dtb.bin 放在U盘 update 目录，或者 tftp 根目录下。 2.uboot菜单选择“Updatedtb“

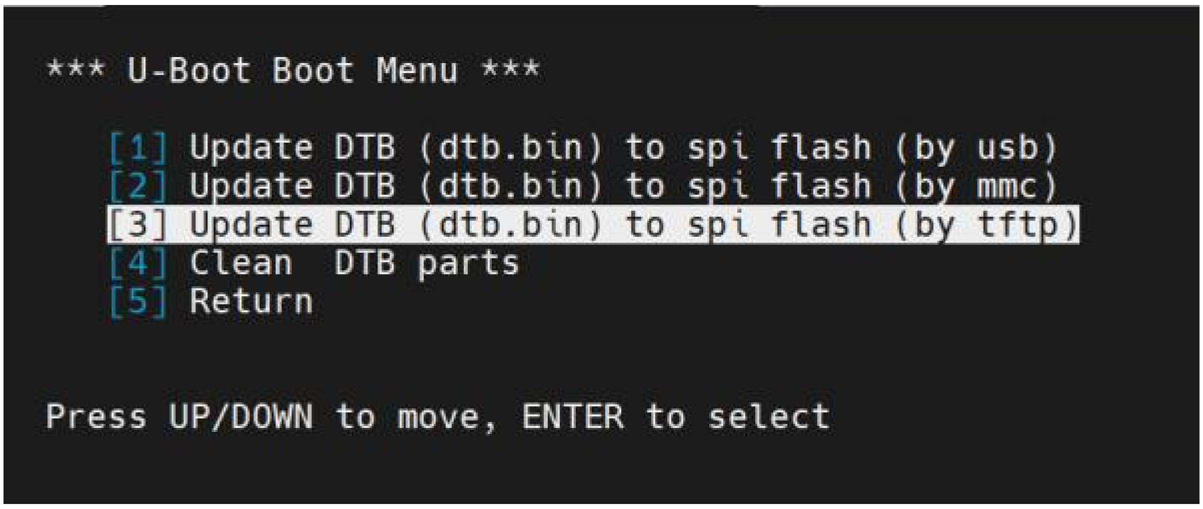


```
*** U-Boot Boot Menu ***

    [1] System boot select
    [2] Update kernel
    [3] Update rootfs
    [4] Update u-boot
    [5] Update dtb
    [6] Update ALL
    [7] System install or recover
    [8] Video resolution select
    [9] Video rotation select
    [a] U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

3.选择“Update DTB (dtb.bin) (by usb)”或“Update DTB (dtb.bin) (by tftp)”，开始升级。

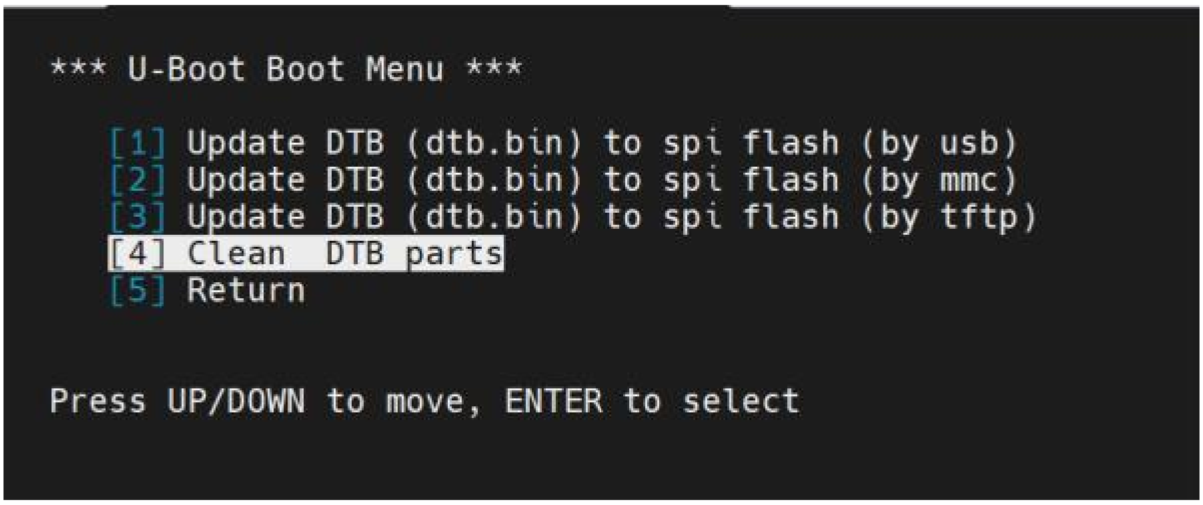


```
*** U-Boot Boot Menu ***

    [1] Update DTB (dtb.bin) to spi flash (by usb)
    [2] Update DTB (dtb.bin) to spi flash (by mmc)
    [3] Update DTB (dtb.bin) to spi flash (by tftp)
    [4] Clean  DTB parts
    [5] Return


Press UP/DOWN to move, ENTER to select
```



```
*** U-Boot Boot Menu ***

    [1] Update DTB (dtb.bin) to spi flash (by usb)
    [2] Update DTB (dtb.bin) to spi flash (by mmc)
    [3] Update DTB (dtb.bin) to spi flash (by tftp)
    [4] Clean  DTB parts
    [5] Return


Press UP/DOWN to move, ENTER to select
```

也可以选择"Clean DTB parts" 使用默认 DTB



```
*** U-Boot Boot Menu ***

    [1] Update DTB (dtb.bin) to spi flash (by usb)
    [2] Update DTB (dtb.bin) to spi flash (by mmc)
    [3] Update DTB (dtb.bin) to spi flash (by tftp)
    [4] Clean  DTB parts
    [5] Return


Press UP/DOWN to move, ENTER to select
```

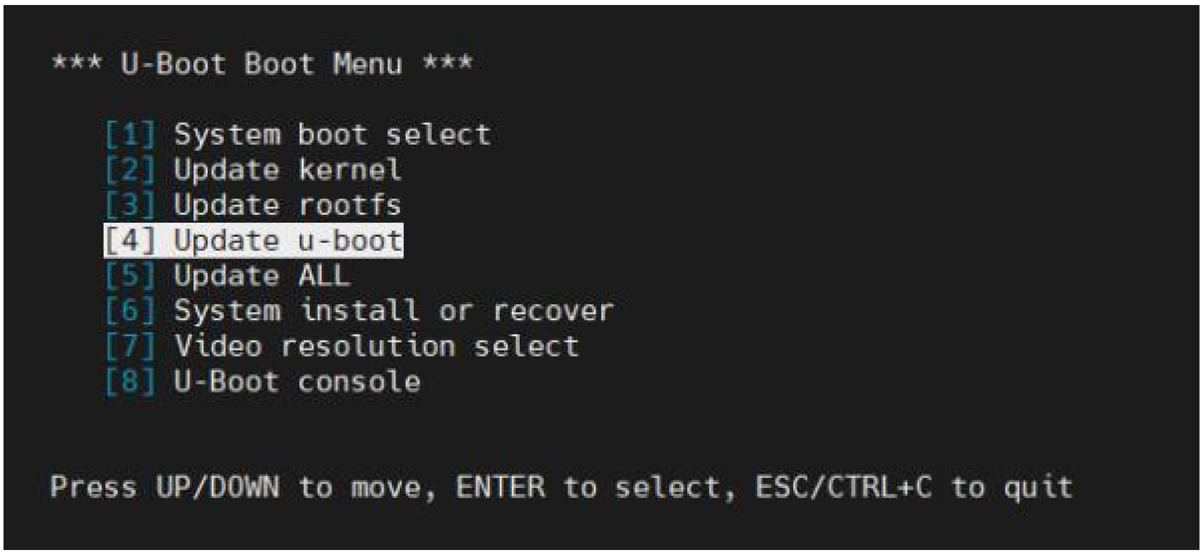
4.升级完成，重启输入 `reboot`或 `reset`

4.7.固件更新

注意：烧录固件需谨慎，可能导致板卡无法启动

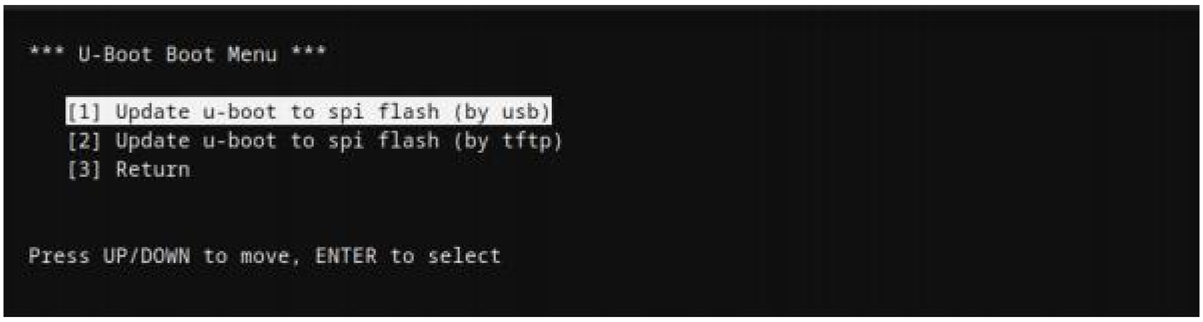
4.7.1.uboot菜单更新固件

1.将 固件（u-boot-with-spl.bin） 放在tftp服务器根目录下，或者U盘update目录下。 1.uboot菜单选择"[4] Update u-boot"

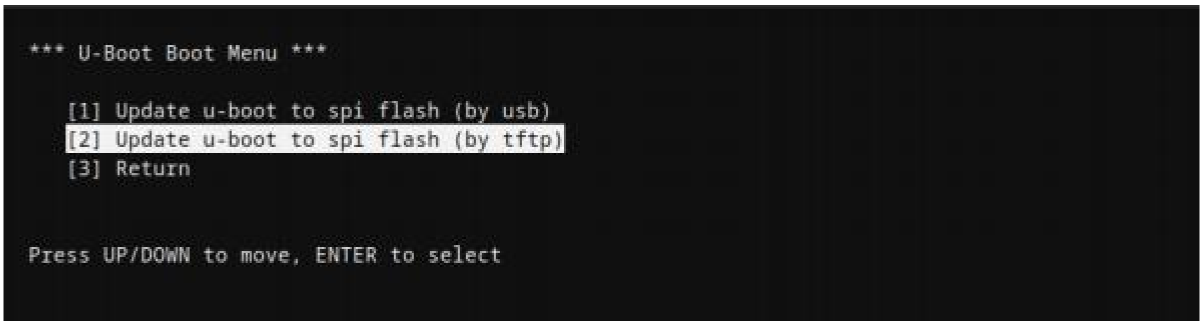


```
*** U-Boot Boot Menu ***

    [1] System boot select
    [2] Update kernel
    [3] Update rootfs
    [4] Update u-boot
    [5] Update ALL
    [6] System install or recover
    [7] Video resolution select
    [8] U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

2.选择"[1] Update u-boot to spi flash (by usb)"或"[2] Update u-boot to spi flash (by tftp)"，开始升 级：

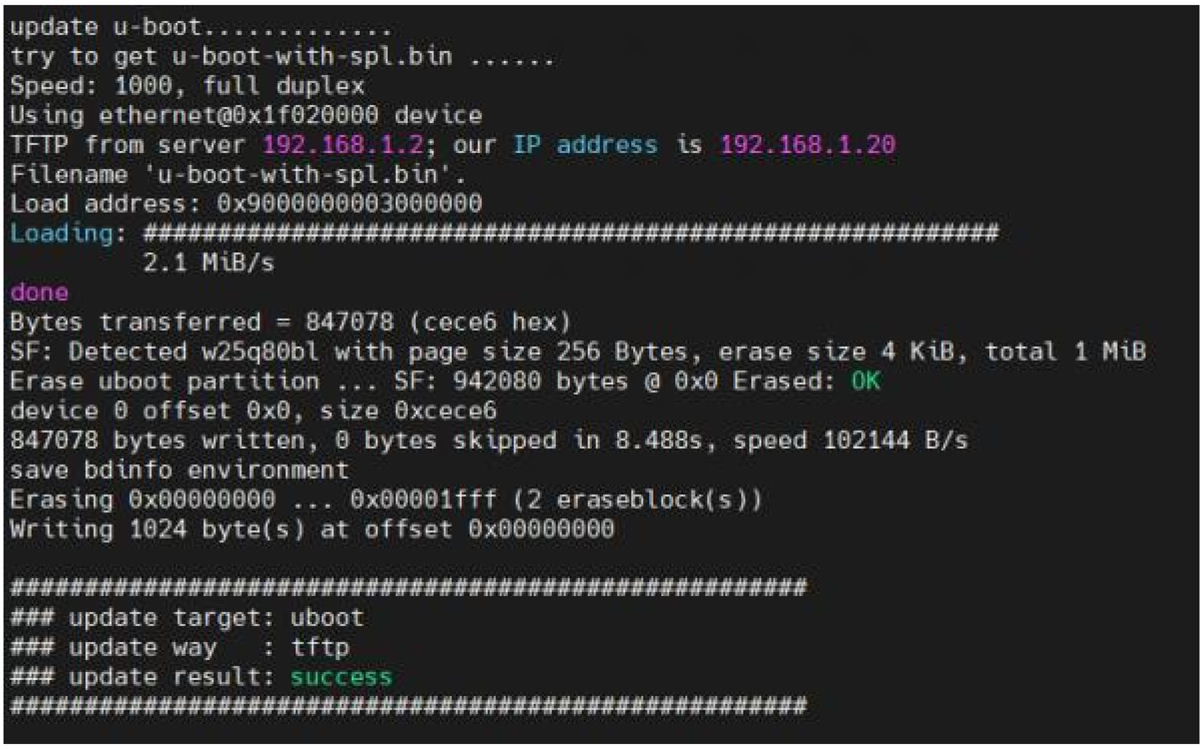


```
*** U-Boot Boot Menu ***

   [1] Update u-boot to spi flash (by usb)
   [2] Update u-boot to spi flash (by tftp)
   [3] Return


Press UP/DOWN to move, ENTER to select
```



```
*** U-Boot Boot Menu ***

   [1] Update u-boot to spi flash (by usb)
   [2] Update u-boot to spi flash (by tftp)
   [3] Return


Press UP/DOWN to move, ENTER to select
```

3.升级完成，重启输入 `reboot`或 `reset`



```
update u-boot.............
try to get u-boot-with-spl.bin ......
Speed: 1000, full duplex
Using ethernet@0x1f020000 device
TFTP from server 192.168.1.2; our IP address is 192.168.1.20
Filename 'u-boot-with-spl.bin'.
Load address: 0x9000000003000000
Loading: ##########################################################
         2.1 MiB/s
done
Bytes transferred = 847078 (cece6 hex)
SF: Detected w25q80bl with page size 256 Bytes, erase size 4 KiB, total 1 MiB
Erase uboot partition ... SF: 942080 bytes @ 0x0 Erased: OK
device 0 offset 0x0, size 0xcece6
847078 bytes written, 0 bytes skipped in 8.488s, speed 102144 B/s
save bdinfo environment
Erasing 0x00000000 ... 0x00001fff (2 eraseblock(s))
Writing 1024 byte(s) at offset 0x00000000

#####################################################
### update target: uboot
### update way    : tftp
### update result: success
#####################################################
```

4.7.2.系统中工具更新固件

更新uboot的软件在 /root/sys_config_tool 目录下，更新步骤如下

1.将 **固件（u-boot-with-spl.bin）**及 **md5校验文件(u-boot-with-spl.bin.md5)** 放在 **sys_config_tool/file**目录下。

2.校验 **sys_config_tool/file** 下固件

```
$ cd sys_config_tool/file $ ls u
-boot-with-spl.bin*
u-boot-with-spl.bin  u-boot-with-spl.bin.md5 $ m
d5sum -c u-boot-with-spl.bin.md5
u-boot-with-spl.bin: OK
```

3. 回到**sys_config_tool** 目录，运行 **update_uboot** 程序，重启

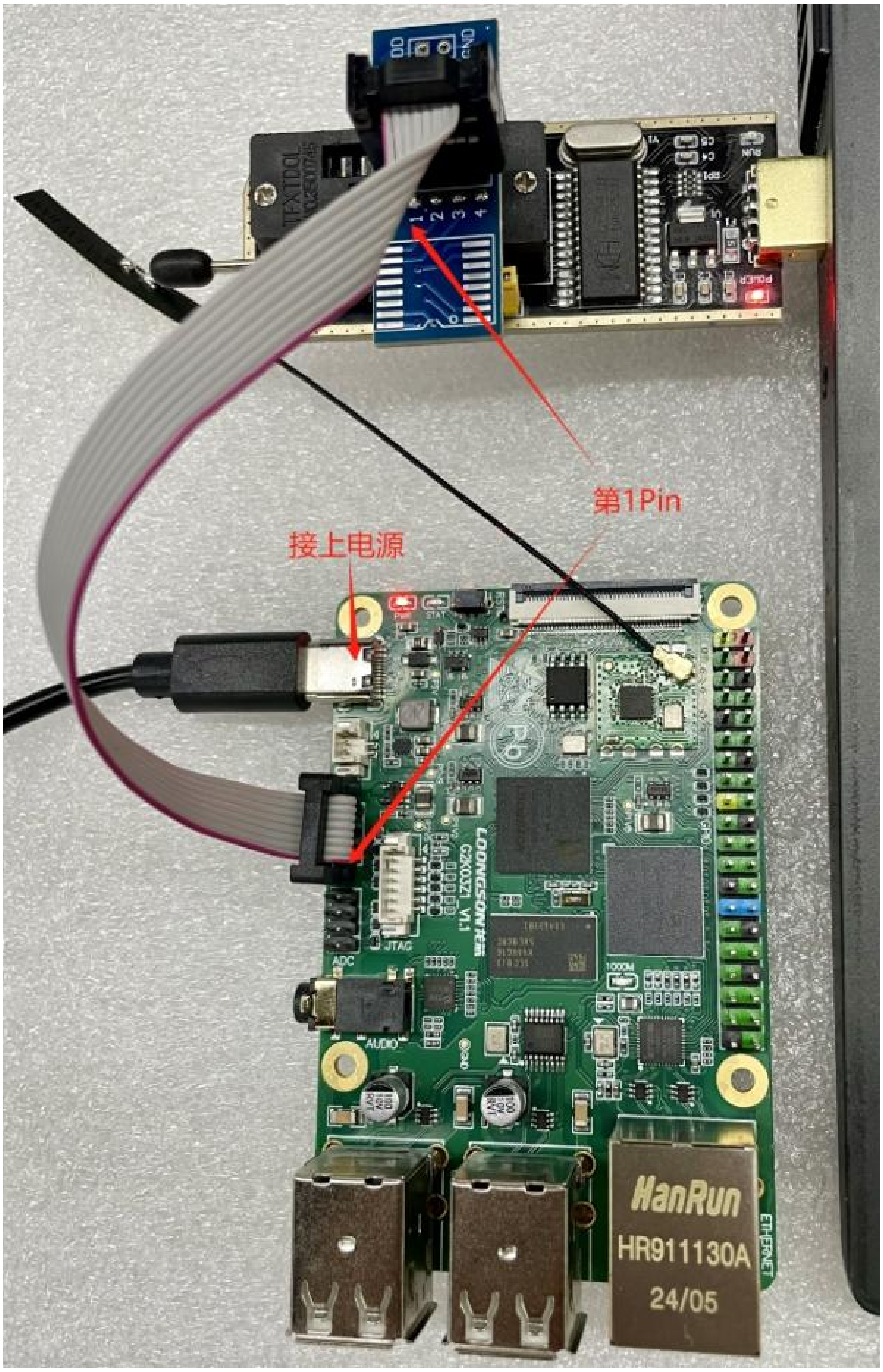
```
$ cd /root/sys_config_tool/file $ ./up
date_uboot
start burn uboot file!
burn uboot file success!
Syncing ...
$ reboot
```

4.7.3.烧录器更新固件

4.7.3.1.烧录器接线

先将蜂鸟板接上电，再将SPI烧录器的一头接在板卡SPI烧录口上（注意线序），一头接在PC机上。连接 方式如图：

> 注意：烧录时建议2K0300蜂鸟板**先接电源**，以避免USB供电不足从而引起烧录失败。


第1Pin
接上电源
LOONGSON
HanRun
HR911130A
24/05
JTAG
AUDIO
ADC
ETHERNET

4.7.3.2.烧录器烧录

烧录有自动与手动两种模式，如自动模式烧录失败，建议采用手动模式烧录。

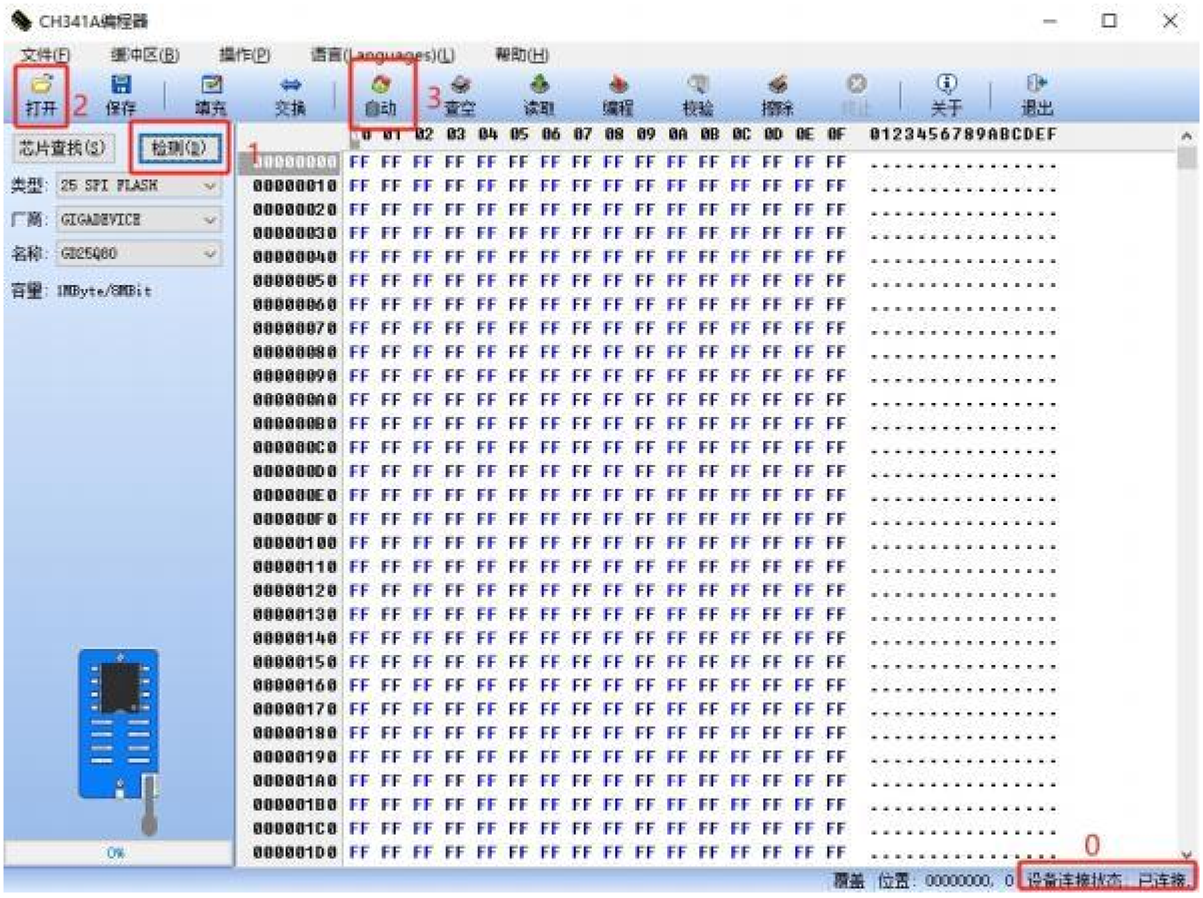
4.7.3.3.自动模式烧录

1.先检查烧录器是否正常被烧录软件识

别 2.点击"检测" 以识别SPI芯片

3.点击"打开" 导入要烧录的固件

4.点击" 自动" 开始烧录，烧录器软件会显示烧录进度

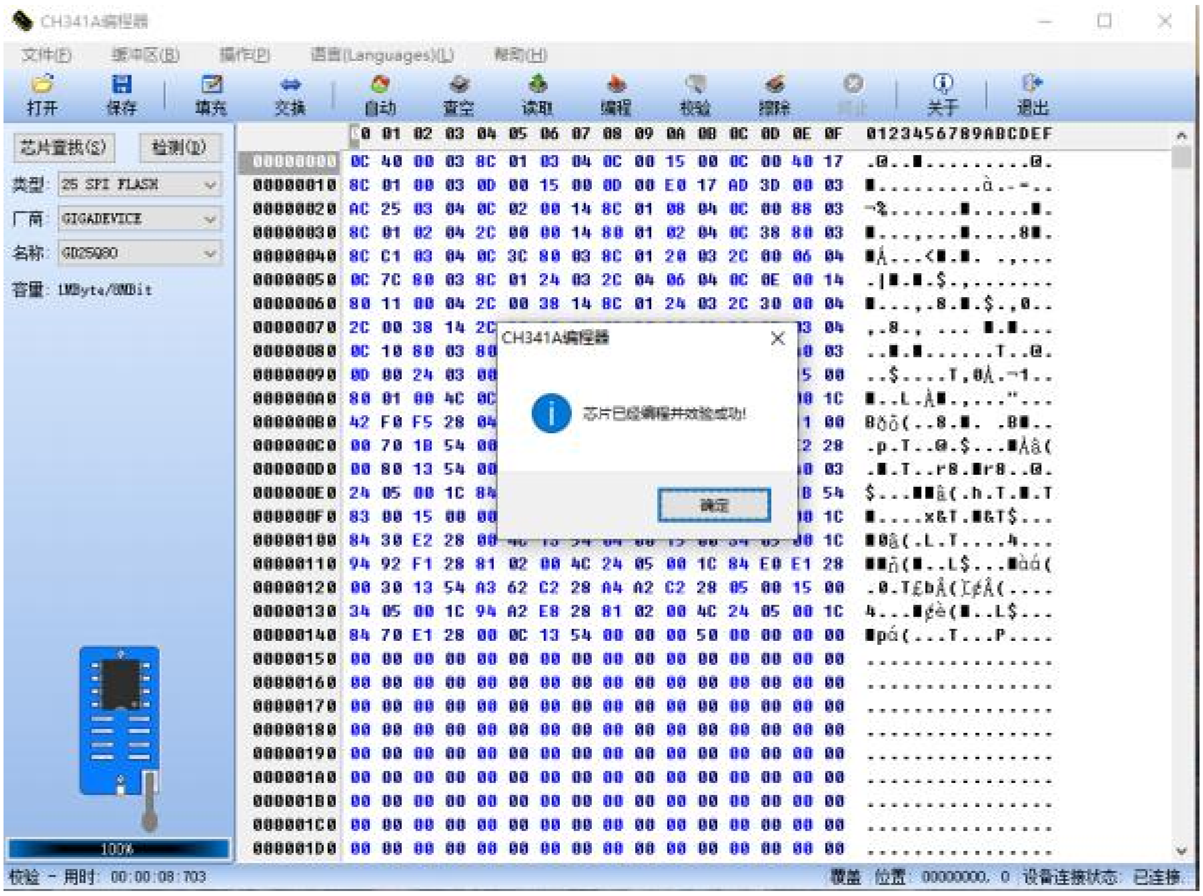


类型: 25 SPI FLASH
厂商: GIGADEVICE
名称: GD25Q80
容量: 1MByte/8MBit

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000030  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000040  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000050  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000060  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000070  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000080  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000090  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000000F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000100  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000110  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000120  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000130  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000140  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000150  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000160  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000170  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000180  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000190  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

覆盖 位置: 0000000, 0 设备连接状态: 已连接.

5.烧录成功，烧录器软件会有弹框提示



类型: 25 SPI FLASH
厂商: GIGADEVICE
名称: GD25Q80
容量: 1MByte/8MBit

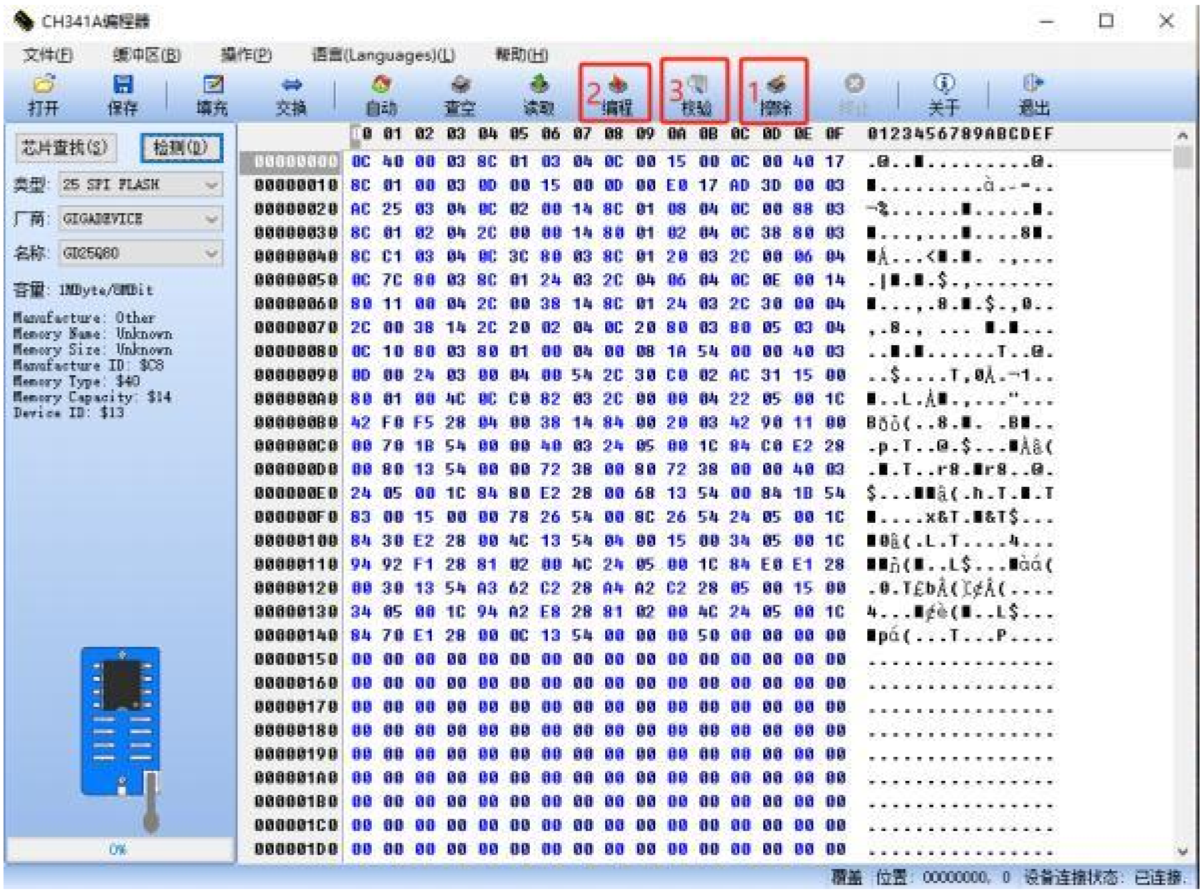
CH341A编程器

芯片已经编程并效验成功!

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000  0C 40 00 03 8C 01 03 04 0C 00 15 00 0C 00 40 17   .@..█.........@.
00000010  8C 01 00 03 0D 00 15 00 0D 00 E0 17 AD 3D 00 03   █.........à.-=..
00000020  AC 25 03 04 0C 02 00 14 8C 01 08 04 0C 00 88 03   ¬%......█.....█.
00000030  8C 01 02 04 2C 00 00 14 80 01 02 04 0C 38 80 03   █...,...█....8█.
00000040  8C C1 03 04 0C 3C 80 03 8C 01 20 03 2C 00 06 04   █Á...<█.█. .,...
00000050  0C 7C 80 03 8C 01 24 03 2C 04 06 04 0C 0E 00 14   .|█.█.$.,.......
00000060  80 11 00 04 2C 00 38 14 8C 01 24 03 2C 30 00 04   █...,.8.█.$.,0..
00000070  2C 00 38 14 2C                            03 04   ,.8., ... █.█...
00000080  0C 10 80 03 80                            40 03   ..█.█......T..@.
00000090  0D 00 24 03 00                            15 00   ..$....T,0Å.¬1..
000000A0  80 01 00 4C 0C                            00 1C   █..L.Å█.,...."...
000000B0  42 F0 F5 28 04                            11 00   Bðõ(..8.█. .B█..
000000C0  00 70 1B 54 00                            E2 28   .p.T..@.$...█Åã(
000000D0  00 80 13 54 00                            40 03   .█.T..r8.█r8..@.
000000E0  24 05 00 1C 84                            1B 54   $...██ã(.h.T.█.T
000000F0  83 00 15 00 00                            00 1C   █....x&T.█&T$...
00000100  84 30 E2 28 00                            00 1C   █0ã(.L.T....4...
00000110  94 92 F1 28 81 02 00 4C 24 05 00 1C 84 E0 E1 28   ██ñ(█..L$...█àá(
00000120  00 30 13 54 A3 62 C2 28 A4 A2 C2 28 05 00 15 00   .0.T£bÂ(¤¢Â(....
00000130  34 05 00 1C 94 A2 E8 28 81 02 00 4C 24 05 00 1C   4...█¢è(█..L$...
00000140  84 70 E1 28 00 0C 13 54 00 00 00 50 00 00 00 00   █pá(...T...P....
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

校验 - 用时: 00:00:08:703   覆盖 位置: 0000000, 0 设备连接状态: 已连接.

4.7.3.4 手动模式烧录



类型: 25 SPI FLASH
厂商: GIGADEVICE
名称: GD25Q80
容量: 1MByte/8MBit

Manufacture: Other
Memory Name: Unknown
Memory Size: Unknown
Manufacture ID: $C8
Memory Type: $40
Memory Capacity: $14
Device ID: $13

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000  0C 40 00 03 8C 01 03 04 0C 00 15 00 0C 00 40 17   .@..█.........@.
00000010  8C 01 00 03 0D 00 15 00 0D 00 E0 17 AD 3D 00 03   █.........à.-=..
00000020  AC 25 03 04 0C 02 00 14 8C 01 08 04 0C 00 88 03   ¬%......█.....█.
00000030  8C 01 02 04 2C 00 00 14 80 01 02 04 0C 38 80 03   █...,...█....8█.
00000040  8C C1 03 04 0C 3C 80 03 8C 01 20 03 2C 00 06 04   █Á...<█.█. .,...
00000050  0C 7C 80 03 8C 01 24 03 2C 04 06 04 0C 0E 00 14   .|█.█.$.,.......
00000060  80 11 00 04 2C 00 38 14 8C 01 24 03 2C 30 00 04   █...,.8.█.$.,0..
00000070  2C 00 38 14 2C 20 02 04 0C 20 80 03 80 05 03 04   ,.8., ... █.█...
00000080  0C 10 80 03 80 01 00 04 00 08 1A 54 00 00 40 03   ..█.█......T..@.
00000090  0D 00 24 03 00 04 00 54 2C 30 C0 02 AC 31 15 00   ..$....T,0Å.¬1..
000000A0  80 01 00 4C 0C C0 82 03 2C 00 00 04 22 05 00 1C   █..L.Å█.,...."...
000000B0  42 F0 F5 28 04 00 38 14 84 00 20 03 42 90 11 00   Bðõ(..8.█. .B█..
000000C0  00 70 1B 54 00 00 40 03 24 05 00 1C 84 C0 E2 28   .p.T..@.$...█Åã(
000000D0  00 80 13 54 00 00 72 38 00 80 72 38 00 00 40 03   .█.T..r8.█r8..@.
000000E0  24 05 00 1C 84 80 E2 28 00 68 13 54 00 84 1B 54   $...██ã(.h.T.█.T
000000F0  83 00 15 00 00 78 26 54 00 8C 26 54 24 05 00 1C   █....x&T.█&T$...
00000100  84 30 E2 28 00 4C 13 54 04 00 15 00 34 05 00 1C   █0ã(.L.T....4...
00000110  94 92 F1 28 81 02 00 4C 24 05 00 1C 84 E0 E1 28   ██ñ(█..L$...█àá(
00000120  00 30 13 54 A3 62 C2 28 A4 A2 C2 28 05 00 15 00   .0.T£bÂ(¤¢Â(....
00000130  34 05 00 1C 94 A2 E8 28 81 02 00 4C 24 05 00 1C   4...█¢è(█..L$...
00000140  84 70 E1 28 00 0C 13 54 00 00 00 50 00 00 00 00   █pá(...T...P....
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

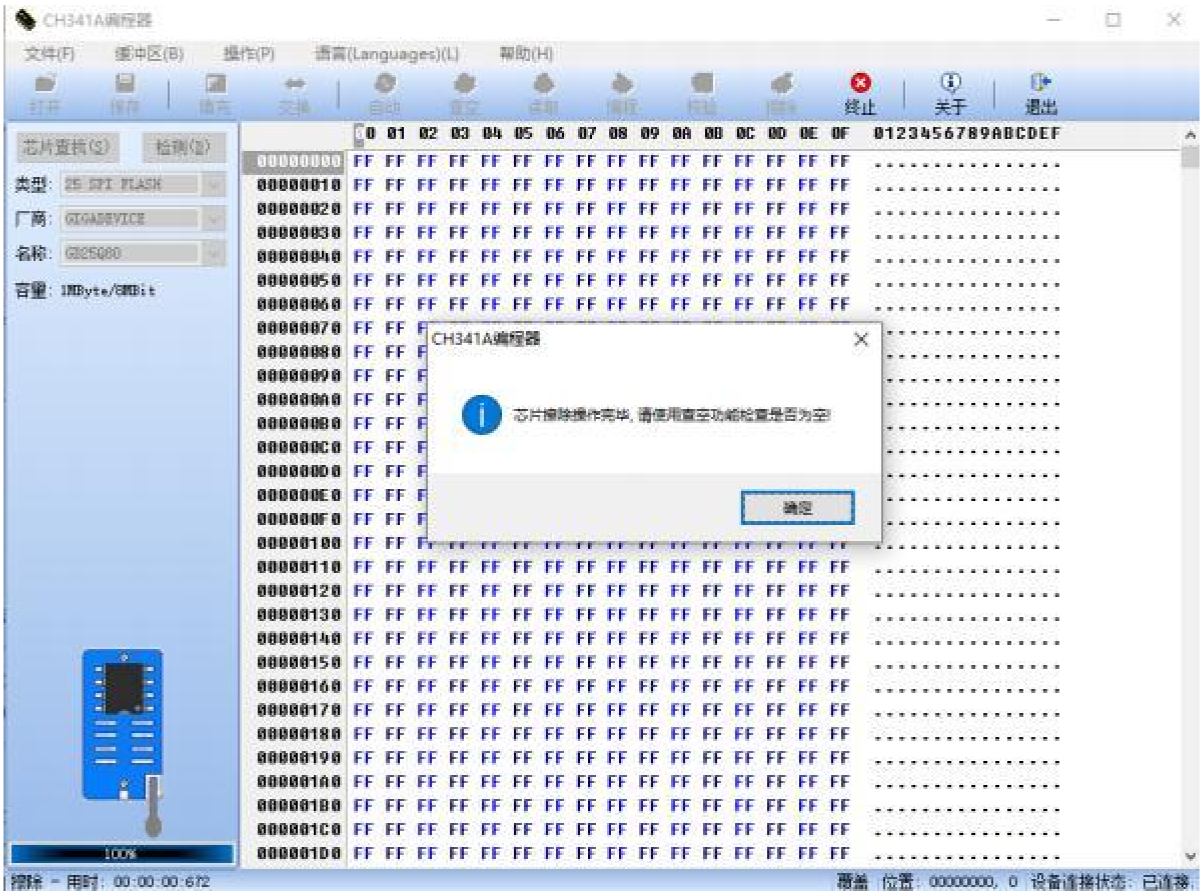
覆盖 位置: 0000000, 0 设备连接状态: 已连接.

1.先检查烧录器是否正常被烧录软件识

别 2.点击"检测" 以识别SPI芯片

3.点击"打开" 导入要烧录的固件

4.点击"擦除" 开始擦除芯片，之后会弹框提示查空，可以忽略。


CH341A编程器
文件(F) 缓冲区(B) 操作(P) 语言(Languages)(L) 帮助(H)
打开 保存 填充 交换 自动 查空 读取 编程 校验 擦除 终止 关于 退出
芯片查找(S) 检测(D)
确定
100%

类型: 25 SPI FLASH
厂商: GIGADEVICE
名称: GD25Q80
容量: 1MByte/8MBit

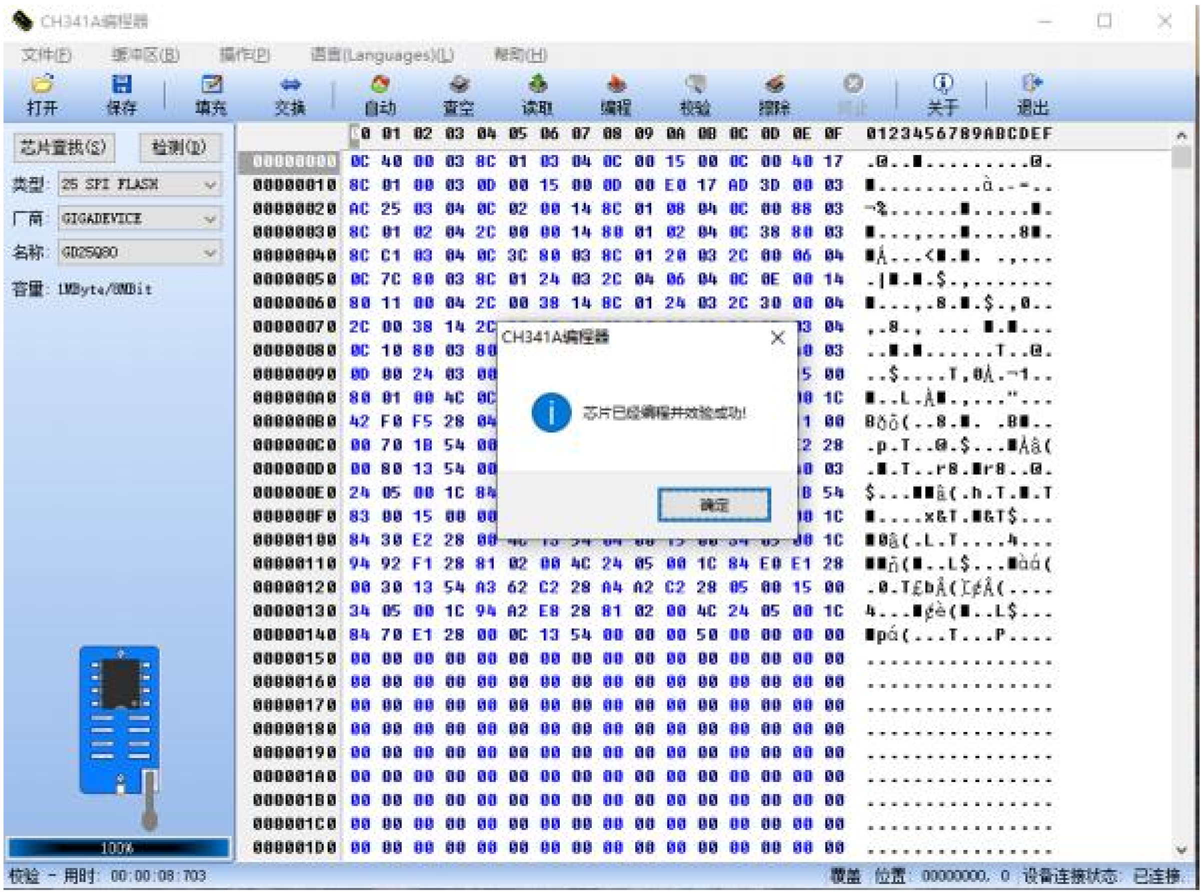
```
         00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000070 FF FF F                                           ................
00000080 FF FF F                                           ................
00000090 FF FF F                                           ................
000000A0 FF FF F                                           ................
000000B0 FF FF F                                           ................
000000C0 FF FF F                                           ................
000000D0 FF FF F                                           ................
000000E0 FF FF F                                           ................
000000F0 FF FF F                                           ................
00000100 FF FF F                                           ................
00000110 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000120 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000130 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000140 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000150 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000160 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000170 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000180 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00000190 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
000001D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

芯片擦除操作完毕, 请使用查空功能检查是否为空!

擦除 - 用时: 00:00:00:672　　覆盖　位置: 00000000, 0　设备连接状态: 已连接

5.点击"编程" 开始烧录

6.待"编程"完成到 100%后，点击"校验" 以确保烧录的正确性。


CH341A编程器
文件(F) 缓冲区(B) 操作(P) 语言(Languages)(L) 帮助(H)
打开 保存 填充 交换 自动 查空 读取 编程 校验 擦除 终止 关于 退出
芯片查找(S) 检测(D)
确定
100%

类型: 25 SPI FLASH
厂商: GIGADEVICE
名称: GD25Q80
容量: 1MByte/8MBit

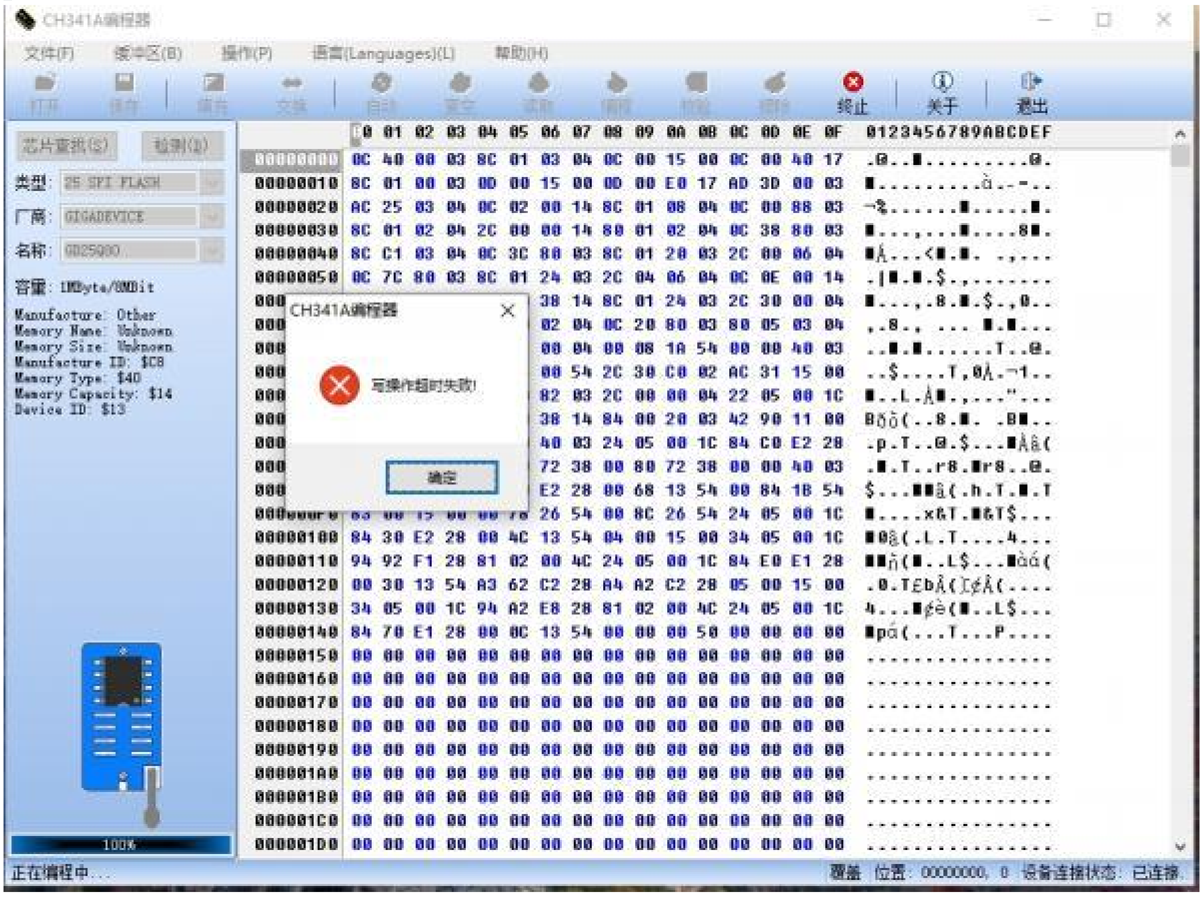
```
         00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   0123456789ABCDEF
00000000 0C 40 00 03 8C 01 03 04 0C 00 15 00 0C 00 40 17   .@..■.........@.
00000010 8C 01 00 03 0D 00 15 00 0D 00 E0 17 AD 3D 00 03   ■..........à.-=..
00000020 AC 25 03 04 0C 02 00 14 8C 01 08 04 0C 00 88 03   ¬%......■.....■.
00000030 8C 01 02 04 2C 00 00 14 80 01 02 04 0C 38 80 03   ■...,...■....8■.
00000040 8C C1 03 04 0C 3C 80 03 8C 01 20 03 2C 00 06 04   ■Á...<■.■. .,...
00000050 0C 7C 80 03 8C 01 24 03 2C 04 06 04 0C 0E 00 14   .|■.■.$.,.......
00000060 80 11 00 04 2C 00 38 14 8C 01 24 03 2C 30 00 04   ■...,.8.■.$.,0..
00000070 2C 00 38 14 2C                         3 04       ,.8., ... ■.■...
00000080 0C 10 80 03 80                         0 03       ..■.■......T..@.
00000090 0D 00 24 03 00                         5 00       ..$....T,0Å.¬1..
000000A0 80 01 00 4C 0C                         0 1C       ■..L.Å■.,...".. 
000000B0 42 F0 F5 28 04                         1 00       Bðõ(..8.■. .8■..
000000C0 00 70 18 54 00                         2 28       .p.T..@.$...■Åã(
000000D0 00 80 13 54 00                         0 03       .■.T..r8.■r8..@.
000000E0 24 05 00 1C 84                         0 54       $...■■ã(.h.T.■.T
000000F0 83 00 15 00 00                         0 1C       ■....x&T.■&T$...
00000100 84 30 E2 28 00                         0 1C       ■0â(.L.T....4...
00000110 94 92 F1 28 81 02 00 4C 24 05 00 1C 84 E0 E1 28   ■■ñ(■..L$...■àá(
00000120 00 30 13 54 A3 62 C2 28 A4 A2 C2 28 05 00 15 00   .0.T£bÂ(¤¢Â(....
00000130 34 05 00 1C 94 A2 E8 28 81 02 00 4C 24 05 00 1C   4...■¢è(■..L$...
00000140 84 70 E1 28 00 0C 13 54 00 00 00 50 00 00 00 00   ■pá(...T...P....
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

芯片已经编程并效验成功!

校验 - 用时: 00:00:08:703　　覆盖　位置: 00000000, 0　设备连接状态: 已连接

注:

> 有时点击" 自动"开始烧录时，会弹出“写超时失败”的弹框，可以忽略此警告，再次点击" 自动"进烧 录即可。



类型: 25 SPI FLASH
厂商: GIGADEVICE
名称: GD25Q80
容量: 1MByte/8MBit

```
Manufacture: Other
Memory Name: Unknown
Memory Size: Unknown
Manufacture ID: $C8
Memory Type: $40
Memory Capacity: $14
Device ID: $13
```

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
00000000  0C 40 00 03 8C 01 03 04 0C 00 15 00 0C 00 40 17  .@..■.........@.
00000010  8C 01 00 03 0D 00 15 00 0D 00 E0 17 AD 3D 00 03  ■..........à.-=..
00000020  AC 25 03 04 0C 02 00 14 8C 01 08 04 0C 00 88 03  ¬%......■.....■.
00000030  8C 01 02 04 2C 00 00 14 80 01 02 04 0C 38 80 03  ■...,...■....8■.
00000040  8C C1 03 04 0C 3C 80 03 8C 01 20 03 2C 00 06 04  ■Á...<■.■. .,...
00000050  0C 7C 80 03 8C 01 24 03 2C 04 06 04 0C 0E 00 14  .|■.■.$.,.......
000                         38 14 8C 01 24 03 2C 30 00 04  ■...,.8.■.$.,0..
000                         02 04 0C 20 80 03 80 05 03 04  ,.8., ... ■.■...
000                         00 04 00 08 1A 54 00 00 40 03  ..■.■......T..@.
000                         00 54 2C 30 C0 02 AC 31 15 00  ..$....T,0À.¬1..
000                         82 03 2C 00 00 04 22 05 00 1C  ■..L.À■.,...."...
000                         38 14 84 00 20 03 42 90 11 00  8õõ(..8.■. .B■..
000                         40 03 24 05 00 1C 84 C0 E2 28  .p.T..@.$...■Àâ(
000                         72 38 00 80 72 38 00 00 40 03  .■.T..r8.■r8..@.
000                         E2 28 00 68 13 54 00 84 1B 54  $...■■â(.h.T.■.T
000000F0  83 00 15 00 00 78 26 54 00 8C 26 54 24 05 00 1C  ■....x&T.■&T$...
00000100  84 30 E2 28 00 4C 13 54 04 00 15 00 34 05 00 1C  ■0â(.L.T....4...
00000110  94 92 F1 28 81 02 00 4C 24 05 00 1C 84 E0 E1 28  ■■ñ(■..L$...■àá(
00000120  00 30 13 54 A3 62 C2 28 A4 A2 C2 28 05 00 15 00  .0.T£bÂ(¤¢Â(....
00000130  34 05 00 1C 94 A2 E8 28 81 02 00 4C 24 05 00 1C  4...■¢è(■..L$...
00000140  84 70 E1 28 00 0C 13 54 00 00 00 50 00 00 00 00  ■pá(...T...P....
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

CH341A编程器

写操作超时失败!

确定

正在编程中...　　覆盖　位置: 00000000, 0　设备连接状态: 已连接

4.7.3 JTAG 烧录固件

龙芯JTAG 仿真器支持在线调试和烧录功能， JTAG 仿真器软件的安装说明见仿真器软件---- >lsdebuger.pdf 文件。

JTAG 仿真器软件支持多种平台，这里以window 系统为例进行使用说明。

解压loongson-debugger-mingw-20240324.tar.xz 到任意目录（建议路径无中文）。进入目录双击 `la32_dbg_tool_usb.exe` 打开软件，会弹出个终端。

在终端输入输入相应的命令即可。具体流程如下：

1.将要烧录的文件（u-boot-with-spl.bin）先负责到JTAG仿真器软件软件目录，后面烧录

需要使用到 2.连接jtag与板卡，再将jtag 仿真器的usb 接到PC上

3.打开 `la32_dbg_tool_usb.exe` ，会弹出命令交互终端 (命令支持Tab 键自动补全)。

4.指定芯片配置 ，在命令交互终端输入 `source configs/config.ls2k300`

5.开发板上电

6.命令交互终端输入 `set`指令，检查是否有寄存器的值返回，有表示连接正常。如果卡住或无返回可以尝试通过复位按钮对板卡进行复位。然后重复步骤2~6

7.命令交互终端输入 `program_cachelock_spi uboot-with-spl.bin` 进行烧录

```
cpu0 -source configs/config.ls2k300  #先指定芯片配置文件
##jtag_clk 1 2
#jtag_clk 8 2
```

```
#if 0x0 ret
#letl clk 8
#expr 0x10000|8
#usblooptest 0x81000070  0x10008
#expr 3>2
#do if 0x1
#letl phase 2
#expr 0x20000|2
#usblooptest 0x81000070  0x20002
#en
d
#ret
#setconfig core.cpucount 1
#setconfig core.cpuwidth 64
#setconfig core.abisize 64
#setconfig helpaddr 0x900000000000f000
#setconfig putelf.uncached 0
#setconfig usb_ejtag.put_speed 0
#setconfig usb_ejtag.get_speed 0
#setconfig core.nocache 1
#setconfig jtag.pcswidth 83
#setconfig jtag.synci 1
#setconfig jtag.jrhb 1
#setconfig jtag.jalrhb 1
#setconfig put.fastdata 0
#setenv ENV_memsize 256
##setenv ENV_highmemsize 1792
#setenv ENV_cpuclock 800000000
#let spibase
0x8000000016010000
#setconfig spi.iobase 0x8000000016010000
#letl ejtag_spibase 0xdb000000000fff00
##setconfig flash.type byte
#setconfig flash.type page
#usblooptest 0x81000070
0x20001 #fix_ejtag_la
#expr1 ( match "/03-soft/loongson-debugger/la32_dbg_tool_usb" .*usb ) == 0
#if 0x0 ret
#usbver
#expr 0x20210129==0x20150105
#do if 0x0
#usblooptest 0x4c
```

```
#expr 0x00001920!=0
#elsif 0x1
#usblooptest    0x40
#usblooptest    0x000019   4
#usblooptest    14 0x44
#usblooptest    0x000019   0
#usblooptest    18 0x48
#usblooptest    0x000019   5
#usblooptest    1c 0x4c
#usblooptest    0x000019   4
#usblooptest    20 0x50
#usblooptest    0x000019   3
#usblooptest    24 0x54
#usblooptest    0x0000192  8
                8
#end
#ret
```

```
#expr "(0x1 << 16)"
#letl PLL_L1_LOCKED      0x10000
#expr "(0x1 << 2)"
#letl PLL_L1_ENA                  0x4
#expr "(0x1 << 2)"
#letl PLL_MEM_ENA        0x4
#expr "(01 << 16)"
#letl PLL_MEM_LOCKED     0x10000
#letl PLL_CHANG_COMMIT 0x1
#letl DDR_REFC    4
#letl DDR_DIV     1
#letl DDR_DIV_L2     4
#letl GPU_DIV_L2     5
#letl PLL_IN 100000000
#expr
333000000/10*1*4*4/(100000000/10)
#letl DDR_LOOPC 0x35
##/* CPU @ 1000Mhz */
#letl L1_LOOPC     80
#letl L1_REFC      4
#letl L1_DIV       1
#letl L2_DIV       2
#expr
1000000000/10*1*4*2/(100000000/10)
#letl L1_LOOPC 0x50
##dc 200M
##gmac 125M
#letl DC_LOOPC     80
#letl DC_REFC      4
#letl DC_DIV       1
#letl DC_DIV_L2    8
#letl GMAC_DIV     16
##letl DC_LOOPC {250000000/10*${DC_DIV}*${DC_REFC}*${DC_DIV_L2}/(${PLL_IN}/10)}
##letl PIX0_LOOPC        109
#letl PIX0_REFC  5
#letl PIX0_DIV   1
#letl PIX0_DIV_L2        20
#expr
64000000/10*1*5*20/(100000000/10)
#letl PIX0_LOOPC 0x40
##letl PIX1_LOOPC        109
```

```
##letl base {($(pci_config_read 0 0x3 0 0x10)&~0xf)|0x9000000000000000}
#source scripts/nand.cmd
#letl nand_msize nand_osize nand_cap  nand_esize 8192 256 7 0x20000
#letl ncmd 0xffffffffbfe06000
#letl orderreg 0xffffffffbfe10c00 #letl
ncmd 0xffffffffbfe06000
#letl orderreg 0xffffffffbfe10c00
#source scripts/spi.cmd
#let spibase
0x8000000016010000 #let spi_cs
0
#let spi_speed 4
#let spibase
0x8000000016010000 #let spi_cs
0
#dellabel gdb_module_setup
##acpi_gmac_suspend()
##{
##devmem 0x4004002c 32 $(($(devmem 0x4004002c)
|0x203)) ##devmem 0x4005002c 32 $(($(devmem
0x4005002c)|0x203))
##devmem 0x1fe0702c  32 0x70
##devmem 0x1fe07028  32
0x0000ffff ##devmem 0x1fe0700c
32 0x0000ffff
##devmem 0x1fe07008 32 $(($(devmem 0x1fe07008)
|0x80)) ##devmem 0x1fe07004 32 $(($(devmem
0x1fe07004)|0x80)) ##devmem 0x1fe07014 32  $(((1<<13)
|(${1:-5}<<10)))
##}
#source scripts/i2c.cmd
#let i2c_noack 0
#source scripts/spi.cmd
#let spibase 0x8000000016010000
#let spi_cs 0
#let spi_speed 4
#let spi_cs 0
#source scripts/dumpserial.cmd
#let kargs 'g console=ttyS0,115200 log_buf_len=10M initcall_debug=1 loglevel=20
```

```
nousb'
#let kernel /tmp/vmlinuz
#let rd /tmp/rootfs.cpio.gz
#let iobase
0x8000000000000000 #info f
name                type  line           contents
gdbaccess           f     48             gdbmap $1 $2 $3 $2  $4
gdb                 f     70             letl m $(setconfig       gdbserver.cpubitmap)
gdb_remote          f     71             letl m $(setconfig       gdbserver.cpubitmap)
gdbmod_remote       f     72             letl m $(setconfig       gdbserver.cpubitmap)
gdbmod              f     73             letl m $(setconfig       gdbserver.cpubitmap)
gdbmod0_remote      f     74             letl m $(setconfig       gdbserver.cpubitmap)
gdbmod0             f     75             letl m $(setconfig       gdbserver.cpubitmap)
eclipse             f     76             letl m $(setconfig       gdbserver.cpubitmap)
eclipse_remote      f     77             letl m $(setconfig       gdbserver.cpubitmap)
eclipsemod          f     78             letl m $(setconfig       gdbserver.cpubitmap)
eclipsemod_remote   f     79             letl m $(setconfig       gdbserver.cpubitmap)
ddd                 f     80             letl m $(setconfig       gdbserver.cpubitmap)
ddd_remote          f     81             letl m $(setconfig       gdbserver.cpubitmap)
dddemod             f     82             letl m $(setconfig       gdbserver.cpubitmap)
dddmod_remote       f     83             letl m $(setconfig       gdbserver.cpubitmap)
fls                 f     189            letl num i "$1" 31
```

```
cache_config        f    198          setconfig core.nocache 1 letl
check_ejtag         f    214          clkdiv ${1:1}
ejtag_check         f    215          letl clkdiv ${1:1}
gdbserver           f    254          letl m $(setconfig gdbserver.cpubitmap)
localmem            f    291          newfunc f_d "devmem \{$$2&0xffffffffff} \{%%d
$$1*8} $$3 $$4"
devmem              f    307          newfunc f_d "echo_2 devmem \
{$$2&0xffffffffff} \{%%d $$1*8}
                                   $$3 $$4"
dummy               f    323          newfunc f_dq "echo_2 $$1 $$2; echo $$RANDOM;"
myput64             f    338          letl f ${1:/srv/tftp/vmlinux}
jtag_clk            f    352          if $(expr1 match "${EJTAGEXE}" .*gpio) ret
rm_jtag_init        f    361          devmem 0x1e6e4008 32 {$(devmem
(3<<30)}                                 0x1e6e4008)|
la64_jtag_init      f    371
(0xb|(5<<25)))|4}                        devmem 0x1fe10444 32 {($(devmem
ls1c_ejtag_init     f    381          0x1fe10444)&~
((0xf<<14)|(1<<21)|(1<<27))}
ls1c_cjtag_init     f    409          devmem 0x1fd011c4 32 {$(devmem
((0xf<<14)|(1<<21)|(1<<27))}
cjtag_wait_reset    f    434          0x1fd011c4)&~ devmem 0x1fd011c4 32
mysi                f    440
mySi                f    444          {$(devmem 0x1fd011c4)&~
fix_ejtag_la        f    448
0) ret
configserial        f    565          do while {$(jtagregs d8q 1 1)!=0x20010819} hb
testserial          f    583          0 -1
set_cpu_clk         f    649          Hb 0 -1
set_ddr_clk         f    659          if $(expr1 ( match "${EJTAGEXE}" .*usb ) ==
set_dc_clk          f    669
set_pix0_clk        f    680          let   serialdiv  ${1:{125000000/115200/16}}
set_pix1_clk        f    691          let   serialdiv  ${1:{125000000/115200/16}}
rtc_set             f    702          letl  t0 0xffffffffbfe10480
${4:12} ${5:0} ${6:0}                 letl   t0 0xffffffffbfe10490
                                      echo dc
rtc_alarmset        f    709          echo pix0
${4:12} ${5:0} ${6:0}                 letl   t0 0xffffffffbfe104c0
rtc_read            f    715          letl y m d h M s ${1:2016} ${2:1} ${3:1}
dm8                 f    723
configddr           f    730          letl y m d h M s ${1:2016} ${2:1} ${3:1}
dumpddr             f    853
ddrtest             f    914          letl c $(d4q {$rtc_reg+0x40} 1) letl i
ddrtest_init        f    915          {$1&0xfff}
ddrtest_uc          f    916          cp0s 0 m8 12 {($(cp0s 0 d8q 12 1)&~4)|0xe0}
erase1              f    935          letl ddrwidth ${1:0}
erase               f    941
program             f    946
```

```
program_bin           f     947
pci_config_read_byte f      964
"$4" "$5"
pci_config_write_byte f      969
"$4" "$5"
pci_config_read       f     979
pci_config_readv      f     980
pci_config_write      f     993
pci_config_read_pci   f     1005
pci_config_read_pciv f     1006
pci_config_write_pci f     1019
```

```
do if $(test $0 == ddrtest_init) do if
$(test $0 == ddrtest_init) do if $(test
$0 == ddrtest_init) #let spi_speed 4

setconfig  spi.iobase  $spibase  letl
file  ${1:/tmp/gzrom.bin}  letl  file
${1:/tmp/gzrom.bin}

letl bus dev func reg cnt  "$1" "$2" "$3"

 let bus dev func reg data  "$1" "$2" "$3"

local bus dev func reg cnt addr addrp q local
bus dev func reg cnt addr addrp q local bus
dev func reg cnt addr type

local bus dev func reg cnt addr addrp q local
bus dev func reg cnt addr addrp q local bus
dev func reg cnt addr type
```

```
select_pci          f     1031     newcmd pci_config_read pci_config_read_pci
pci_find_cap        f     1047     letl bus dev func type ${1:0} ${2:0} ${3:0}
${4:0x10}
pcie_max_dev        f     1058     letl bus dev func ${1:0} ${2:0} ${3:0}
pci_list_bus        f     1072     letl show $(setconfig sys.showcmd)
pci_playload        f     1104     letl bus dev func  ${1:0} ${2:0} ${3:0}
bus2phys            f     1116     echo $1
bus2virt            f     1119     echo {$1|0x9000000000000000}
dumpe1000e          f     1122     letl bus dev func ${1:0} ${2:0} ${3:0}
reboot_tos          f     1153     set_gpio_out 46 1
reboot1             f     1158     memset4 0x800000001ff6c030 1 4
reboot2             f     1161     letl v $(d4q 0x8000000016010204 1)
reboot              f     1166     m4 0x800000001ff6c030 2
reboot_off          f     1171     m4 0x800000001ff6c014 0x3c00
poweroff            f     1174     m4 0x800000001ff6c00c $(d4q
0x800000001ff6c00c 1)
i2c0_init           f     1179     let i2creg {0xffffffffbfe01000+${1:0}*0x800};
i2c0_read           f     1186     letl cr dr adr reg count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
i2c0_write          f     1209     letl cr dr adr reg val $(expr $i2creg+0x4)
$(expr $i2creg+0x3) ${1:0} ${2:0} ${3:0}
i2c0_read2a         f     1224     letl cr dr adr reg count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
i2c0_write2a        f     1250     letl cr dr adr reg val $(expr $i2creg+0x4)
$(expr $i2creg+0x3) ${1:0} ${2:0} ${3:0}
i2c0_read2          f     1268     letl cr dr adr reg count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
i2c0_write2         f     1296     letl cr dr adr reg val $(expr $i2creg+0x4)
$(expr $i2creg+0x3) ${1:0} ${2:0} ${3:0}
net                 f     1314     let gmacreg $(pci_config_read 0 3 0 0x10 1)
exectest            f     1322     letl p 0xffffffff90000000
exectest            f     3436     letl p 0xffffffff90000000
cache_init          f     1333     setconfig core.nocache 0
cache_lock          f     1341     m8 0x8000000016000200  0x800000001c000000
cache_init_and_lock  f     1345     call cache_init
cache_init_and_lock1 f     1346     call cache_init
memtest_cachelock    f     1357     echo_on
program_cachelock_spi f     1369     spi_memen
program_cachelock    f     1370     spi_memen
fprogram_cachelock_spi f     1372     spi_memen
program_cachelock_bin f     1373     spi_memen
get_cachelock       f     1415     setconfig helpaddr 0x900000001c000000
get_cachelock       f     3104     setconfig helpaddr 0x900000001c000000
reset_lpc           f     1427     letl v $(d4q 0xffffffffbfd00200 1)
```

```
enable_pcieclk      f     1433     m4 0xffffffffbfe10430 {$(d4q
0xffffffffbfe10430 1)|0x30000}
disable_pcieclk     f     1436     m4 0xffffffffbfe10430 {$(d4q
0xffffffffbfe10430 1)&~0x30000}
testpcie            f     1439     letl portnum ${1:0}
init_pcieport       f     1456     letl port ${1:0xe}
trainen_pcieport    f     1471     letl port ${1:0xe}
forcestate_pcieport f     1488     letl port ${1:0xe}
init_pciephy        f     1498     m4 0xffffffffbfe10580 0xc2492331
pcie_train          f     1514     letl port ${1:0xe}
pcie_lookback       f     1528     m1 0xb8110011 0x2
read_phy            f     1536     @letl show $(@setconfig sys.showcmd)
```

```
write_phy              f      1552                           @letl show $(@setconfig sys.showcmd)
write_phy_reg_val      f      1563                   let phy ${1:0}
read_phy_reg_cnt       f      1567                   let phy ${1:0}
print_speed            f      1571                   letl stat $1
print_speed1           f      1598                   letl s $(d4q {$base+0xd8} 1)
find_phy               f      1620                   letl phy0 -1
switch_mv88e6070_phy_read f         1644      @echo_off
switch_mv88e6070_phy_write f         1656      @echo_off
vbus2phys                    f         1664           expr ($1>0x90000000)?$1:($1&0x0fffffff)
gmacdesc                     f         1667       letl bar0 $base
read_switch                  f         1711       letl oldphy $phy
write_switch                 f         1723       letl oldphy $phy
config_switch                f         1738       write_switch 0x8 0x1c
set_gpio_out                 f         1748       letl pin val ${1:0} ${2:0}
physaddr                     f         1765       echo
{($1>0xffffffff80000000&&$1<0xffffffff90000000)?($1&0x1fffffff):($1&0xffffffff)}
nand_init              f      1768      m4 0xffffffffbfe10420 {$(d4q
0xffffffffbfe10420 1)|(1<<9)}
nand_reset             f      1771      letl naddr {$ncmd+8}
nand_readid            f      1791      letl naddr {$ncmd+8}
nand_erase             f      1828      letl naddr {$ncmd+8}
nand_Erase             f      1854      letl naddr {$ncmd+8}
nand_read              f      1874      letl naddc {$ncmd+4}
nand_write             f      1915      letl naddc {$ncmd+4}
spi_set_cs           f      1962      do if $1
spi_set_cs           f      2910      do if $1
spi_send_data          f      1969      m1 {$spibase+0x2} $1
spi_send_data          f      2917      m1 {$spibase+0x2} $1
spi_qsend_data         f      1970      m1 {$spibase+0x2} $1
spi_qsend_data         f      2918      m1 {$spibase+0x2} $1
spi_readid             f      1976      call spi_init $spi_speed
spi_readid             f      2924      call spi_init $spi_speed
spi_read_area          f      1986      let addr cnt ${1:0} ${2:16}
spi_read_area          f      2934      let addr cnt ${1:0} ${2:16}
spi_wait_sr            f      2005      spi_set_cs 0
spi_wait_sr            f      2953      spi_set_cs 0
spi_set_wren           f      2011      spi_wait_sr
spi_set_wren           f      2959      spi_wait_sr
spi_write_sr           f      2021      let sr ${1:0}
spi_write_sr           f      2969      let sr ${1:0}
spi_read_sr            f      2030      spi_set_cs 0
spi_read_sr            f      2978      spi_set_cs 0
spi_write_area         f      2036      let addr ${1:0}
spi_write_area         f      2984      let addr ${1:0}
spi_init             f      2053      letl d ${1:4}
spi_init            f    3001    letl d ${1:4}
spi_mytest          f    2063    call spi_init $spi_speed
spi_mytest          f    3011    call spi_init $spi_speed
spi_memen           f    2073    letl en ${1:1}
spi_memen           f    3021    letl en ${1:1}
spi_sst_wen         f    2082    letl en {${1:1}?0:0xff}
spi_sst_wen         f    3030    letl en {${1:1}?0:0xff}
spi_winbond_wen     f    2098    call spi_init $spi_speed
spi_winbond_wen     f    3046    call spi_init $spi_speed
spi_erase_all       f    2106    call spi_init $spi_speed
spi_erase_all       f    3054    call spi_init $spi_speed
```

```
spi_erase_area       f     2119              let addr ${1:0}
spi_erase_area       f     3067              let addr ${1:0}
spi_gd25_diesel      f     2132              call spi_init $spi_speed
spi_gd25_diesel      f     3080              call spi_init $spi_speed
spi_cmd              f     2139              spi_set_cs 0
spi_cmd              f     3087              spi_set_cs 0
spi_fifoclear        f     2150              do while {($(d1q {$spibase+0x1} 1)&1)==0}
spi_fifoclear        f     3098              do while {($(d1q {$spibase+0x1} 1)&1)==0}
vxworks0             f     2157              setconfig core.abisize 32
vxworks              f     2164              setconfig core.abisize 32
vxworks1             f     2173              setconfig core.abisize 32
stfillbuffer_disable f     2180              letl v {$(cp0s 6 d4q 16 1)&~0x100}
config_ls2k_xbar     f     2184              m8 0xffffffffbfe10020 0;m8 0xffffffffbfe10060
0;m8 0xffffffffbfe100a0 0xf2
gdb_module_setup     f     2188              >> gdb.cmd echo monitor python pgdorder=0
acpi_suspend         f     2203              letl acpibase 0xffffffffbff6c000
rtc_wake             f     2218              letl t ${1:10}
wdt_close            f     2224              set_gpio_out 3 0
cachedump            f     2227              setconfig putelf.uncached 2
cachedump1           f     2233              setconfig putelf.uncached 2
lio_config           f     2241              letl width8 speed ${1:0} ${2:0}
lio_cmd              f     2247              do while {$#>1}
lio_id               f     2253              lio_cmd 0x555 0xaa 0x2aa 0x55 0x555 0x90
pr2_func             f     2258              #lsu-clk 25MHz cfg
pr4_func             f     2331              #lsu-clk 25MHz cfg
lsu_st               f     2437              m4 0xffffffffbf0c0240 0x100
lsu_end              f     2443              m4 0xffffffffbf0c0240 0x1000
jbig_func            f     2448              m4 0xffffffffbf0c201c 0x01
runlinux             f     2457              inputserial "c\b" 0xffffffffbff40805
0xffffffffbff40800
test_pciaccess       f     2466              call cache_init
test_pcidma          f     2472              letl cfg $(d4q 0x8000000016000110 1)
erase_spi            f     2515              setconfig spi.iobase $spibase
program_spi          f     2522              letl file ${1:/tmp/gzrom.bin}
mmc_probe            f     2534              letl file ${1:/tmp/gzrom-mmc.bin}
mmc_vmlinux          f     2553              letl file ${1:/tmp/vm.2p.jpeg}
fixup_window         f     2572              m4 0x8000000016000100 {$(d4q
0x8000000016000100 1)|0x20}
i2c_delay            f     2577              #msleep $1
i2c_gpio_start       f     2585              set_gpio_out $gpio_d 1
i2c_gpio_send        f     2592              let d $1
i2c_gpio_rec         f     2605              letl d $(get_gpio_in $gpio_d)
i2c_gpio_stop        f     2621              set_gpio_out $gpio_c 1
i2c_gpio_rack        f     2627              letl d $(get_gpio_in $gpio_d)
i2c_gpio_wack        f     2635              set_gpio_out $gpio_d $1
i2c_gpio_read        f     2640              letl addr reg ${1:0} ${2:0}
i2c_gpio_write       f     2661              letl addr reg dat ${1:0} ${2:0} ${3:0}
i2c_via_read         f     2679              #i2c_init
i2c_via_read_block   f     2699              #i2c_init
```

```
i2c_via_read_byte    f     2720              letl device data ${1:0} ${2:0}
i2c_via_write_byte   f     2733              #i2c_init
_i2c_ls2k_stop       f     2747              letl cr dr $(expr $i2creg+0x4) $(expr
$i2creg+0x3)
_i2c_ls2k_read       f     2752              letl cr dr adr count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "${2:1}"
```

```
_i2c_ls2k_write           f       2765      letl cr dr adr $(expr $i2creg+0x4) $(expr
$i2creg+0x3) ${1:0}
i2c_ls2k_write            f       2784      _i2c_ls2k_write $*
i2c_ls2k_read             f       2788             letl cr dr adr reg count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
i2c_ls2k_scan             f       2798      local r
pci8619_i2c_read          f       2809      letl adr reg  port      {${3:0x38}<<1} ${1:0x78}
${2:4}
i2c_ls2k_read2a           f       2816      letl cr dr adr reg         count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
i2c_ls2k_write2a     f     2842     letl cr dr adr reg              val $(expr $i2creg+0x4)
$(expr $i2creg+0x3) ${1:0} ${2:0} ${3:0}
i2c_ls2k_read2       f     2860     letl cr dr adr reg                   count $(expr $i2creg+0x4)
$(expr $i2creg+0x3) "$1" "$2" $(expr ($#>3)*($3-1)+1)
i2c_ls2k_write2      f            2888      letl cr dr adr reg val $(expr $i2creg+0x4)
$(expr $i2creg+0x3) ${1:0}      ${2:0}   ${3:0}
dumpserial           f            3118      mems
dumpserial.hb        f            3130      letl saddr ${1:0xffffffffbfe001e0}
watchprint           f            3150      echo_off
awatchprint          f            3169      echo_off
mywatchset           f            3195      echo_off
mywatch              f            3229      echo_off
mywait               f            3268      do while 1
watchgpu             f            3274      echo_off
rwatchpcicfg         f            3301      echo_off
watchddrparam          f          3324      echo_off
inputserial          f            3346      setconfig log.level 0
waitnmi              f            3378      do while 1
myput                f            3391             #myput serialaddr #use let to set kernel, rd,
kargs to boot kernel
waitserial          f     3404     letl str ${1:PMON>}
#echo_on
cpu0 -
cpu0 -set   #通过set 命令可以查看寄存器的状态
#set

zero:0x0                   ra:0x9000000000e319c   tp:0x90000000011dc00
                           c                      0
sp:0x90000000011dfdc0
  a0:0x90000000011fea8     a1:0x900000000012021   a2:0x1                 a3:0xa
  0                        00
  a4:0x3                   a5:0xffffffffffffffff  a6:0x8080808080808080  a7:0x16
                                                  80
  t0:0x0                   t1:0x4                 t2:0x14547ee
t3:0xf5c28f
  t4:0xffffffffffffffff    t5:0x9ab0b8e2c90       t6:0x90000000011ee29
                                                  0
t7:0x90000000011ee290
  t8:0x53c17e1705106d7     u0:0x0                 fp:0x200000            s0:0x4
  a
  s1:0x0                   s2:0x90000000011ee41   s3:0x4
                           4
```

```
s4:0x90000000012aca5f
   s5:0x900000000010888   s6:0x1                          s7:0x900000000eca00a   s8:0x0
   28                                                     8
   pc:0x90000000002018
   c0

csr0-crmd:0xb4               csr1-prmd:0x4                                csr2-cu   :0x0
                   csr3-cfg  :0x0
csr4-excfg:0x71c1c           csr5-exst:0x800                              csr6-epc
:0x90000000002018c0    csr7-badv:0x7ffbe3b0ca
csr8-badi:0x0                csrc-ebase:0x9000000001700000
cpu0  -
```

```
cpu0 -program_cachelock_spi u-boot-with-spl.bin   #烧录固件
#program_cachelock_spi u-boot-with-spl.bin
#spi_memen
#letl en 1
#expr
0x8000000016010000+0x4
#d1q 0x8000000016010004
1
#letl o 0x47
#do if 1
#expr
0x8000000016010000+0x4
#expr 0x47|1
#m1  0x8000000016010004
0x47 #loop_break
#ret
#letl file u-boot-with-spl.bin
#echo_on
#stop
#setconfig usb_ejtag.put_speed 0x400
#setconfig callbin.stacksize 0x1000
#setconfig helpaddr
0x900000001c000000 #setconfig
put.pack_size 0x10000
#call cache_init_and_lock
#call cache_init
#setconfig core.nocache 0
#setconfig cacheflush.nohelp_size 0x1000000 #set
csr0 0xb0
#set csr180
0x800000000000000f #set
csr181 0x900000000000001f
#setconfig core.nocache 1
#ret
#call fixup_window
#d4q 0x8000000016000100 1
#expr 0x18020010|0x20
#m4                          0x18020030
0x8000000016000100
#m8                                              0x000000001c0000f2
0x8000000016000090                               0x000000001c0000f2
#ret
#call cache_lock
#m8                           0x800000001c000000
0x8000000016000200
```

```
#m8                              0xffffffffffc0000
0x8000000016000240
#ret
#smemset8 0x900000001c000000 0 0x800
...

#do while 0x1#put u-boot-with-sp1.bin 0x900000001c010000 0x20000
0xc0000 pack:1,time :5,domload size :0xld332,dowmload rate=23920 B/S
#setconfig 1og.disas1
#1og /tmp/1og-$i.txt
#test program cachelock spi ==program cachelock bin #do
if 0
#Sprogram 0x900000001c0100000xc0000
1196020x00000000000c0000(786432)0x00000000000d000
0(851968) #end
##let1 s s
##setconfig 1og.disas 0
##log
#end
#expr0xc0000+0x20000
```

```
#let1 i 0xe0000
#expr
0xe0000<906034 #do
whi1e 0x0
#setconfig helpaddr
0x900000000000f000 #setconfig
```

五、开发板文件系统使用

本章节着重讲解2k300开发板上文件系统使

用。 本章节包含的内容包括：

1. 文件系统所用的文件系统的类型
2. 文件系统的安装方式
3. 文件系统中的软件编译方式
4. 文件系统的制作

如果想了解更多关于buildroot的新架构适配和最小系统的构建。可以查看 9.1.buildroot 章节。

请注意，本章的内容属于文件系统这一种通用功能的介绍文本。下文中的开发板即指代各型号广州龙芯的嵌入式开发板。

开发板上可以部署文件系统到EMMC或TF卡中，均是**ext4**类型，文件系统的部署包的名字默认是 **rootfs.img**。

EMMC的规格是随板卡出厂，存储大小默认为8G。TF卡建议存储大小也维持在8GB及以上。

5.1.查看系统信息

显示操作系统的内核版本号

```
# uname -a
Linux LS-GD 5.10.0.lsgd+ #1 PREEMPT g210c2be51 Fri Sep 6 10:53:58 CST 2024
loongarch64 GNU/Linux
```

查看系统主机名

```
# cat /etc/hostname
LS-GD
```

查看系统登录开机信息，（备注：非自动登录时会打印开机信息）

```
# cat /etc/issue
Welcome to Loongson-gd
```

查看 CPU 相关信息

```
# cat /proc/cpuinfo
system type                     : generic-loongson-machine
processor                  : 0
package                    : 0
core                       : 0
CPU Family                 : Loongson-64bit
Model Name                 :
CPU Revision               : 0x30
FPU Revision               : 0x00
CPU MHz                    : 1000.00
BogoMIPS                   : 2000.00
```

```
TLB Entries                 : 64
Address Sizes               : 40 bits physical, 40 bits virtual
ISA                         : loongarch32 loongarch64
Features                    : cpucfg lam fpu
Hardware Watchpoint              : yes, iwatch count: 4, dwatch count: 2
```

查看内存相关信息

```
# cat /proc/meminfo

MemTotal:         384144 kB
MemFree:          186512 kB
MemAvailable:     283024 kB
Buffers:            6304 kB
Cached:            97456 kB
SwapCached:            0 kB
Active:            25424 kB
Inactive:         131440 kB
Active(anon):        816 kB
Inactive(anon):    56208 kB
Active(file):      24608 kB
Inactive(file):    75232 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:               144 kB
Writeback:             0 kB
AnonPages:         53248 kB
Mapped:            60160 kB
Shmem:              3936 kB
KReclaimable:       6496 kB
Slab:              20224 kB
SReclaimable:       6496 kB
SUnreclaim:        13728 kB
KernelStack:        1392 kB
PageTables:         2864 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:      192064 kB
Committed_AS:     217488 kB
VmallocTotal:   532348896 kB
VmallocUsed:        1968 kB
VmallocChunk:          0 kB
Percpu:              288 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
```

```
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:      32768 kB
```

5.2. LED控制

有用个LED 为心跳灯，用于指示系统运行。
进入开发板系统，在串口终端执行指令控制对应的 IO 来控制对应的器件：

开发板上启动后 LED 默认是[heartbeat]模式，执行如下指令改变当前触发模式，改成[none]模式。

改变 LED 的触发模式

```
# echo none > /sys/class/leds/led1/trigger
```

点亮 LED

```
# echo 1 > /sys/class/leds/led1/brightness
```

熄灭 LED

```
# echo 0 > /sys/class/leds/led1/brightness
```

5.3.CPU内部温度传感器读写

有两种方法获取cpu温度

。 方法1：

```
# cat /sys/class/thermal/thermal_zone0/temp

47000
```

将得到数值除以 1000 即是CPU的内部的温

度。 方法2：

```
# sensors
cpu_thermal-virtual-0

Adapter: Virtual device

temp1:        +47.2 C  (crit = +125.0 C)
```

对于三种类型的文件系统（loongnix带界面版和字符串版， busybox系统）。对于loongnix系统的使 用，只需要像使用PC机那样进行操作即可。由于带有编译系统，所以可以将源码放置于系统中进行编 译，下文不再赘述。

5.4.修改开机执行自定义动作

busybox系统使用了systemd作为启动进程。关于systemd的相关描述本文不再赘述。

而系统中，在/root/下的boot_run.sh脚本则是类似于/etc/profile的作用。可修改此脚本指定系统启动后的动作。这个脚本的执行依靠于/usr/lib/systemd/system/boot_run.service的Exec_start字段。关于 systemd的服务文件编写规则，本文不再赘述。

如果想关闭此服务，即开机不执行boot_run.sh脚本，那么请输入命令：

```
systemctl disable boot_run
```

如果想重新打开此服务，开机启动boot_ru.sh脚本，那么请输入命令：

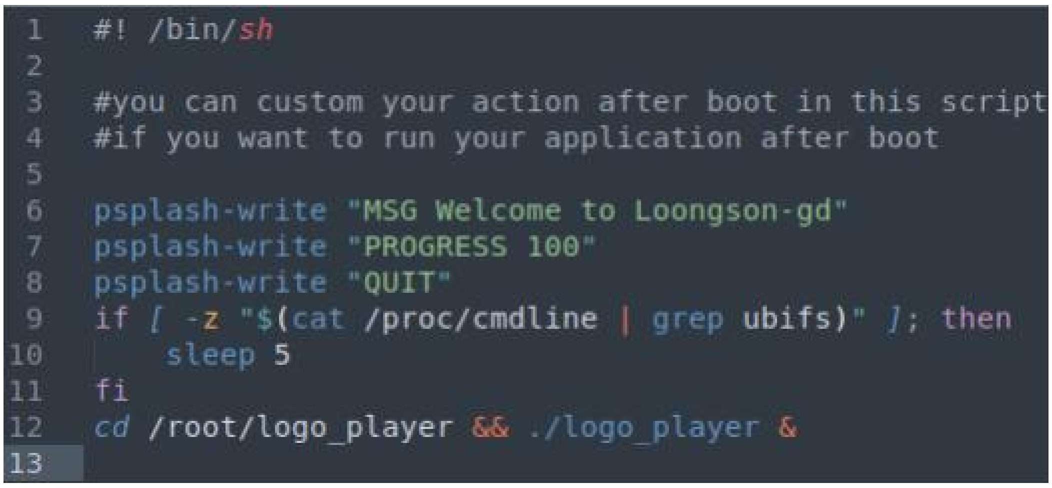
```
systemctl enable boot_run
```

事实上，在/etc/systemd/system/multi-user.target.wants下面，会有一个链接文件，为
boot_run.service。链接的路径正是/usr/lib/systemd/system/boot_run.service。其实上述的两条命令 只不过是删除或者添加此链接而已。

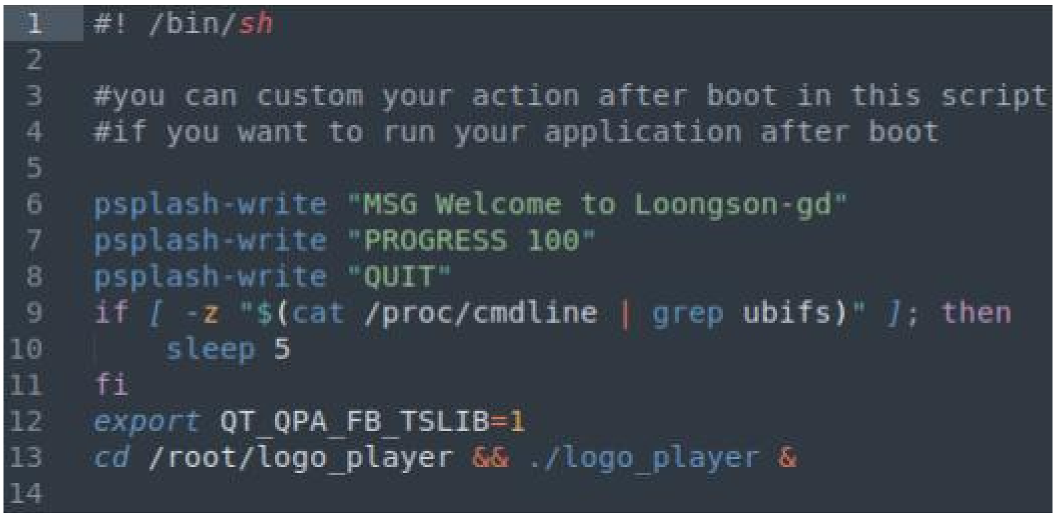
boot_run.service的type为forking，那么如果启动图形程序，命令中是可以使用&，后台运行的。并且 更加推荐使用&，不然图形程序会卡住boot_run .service无法退出，可能会影响systemd-analyze等软件 的使用。

目前提供的busybox系统中见下面两张图，为boot_run.sh的内容。

下面的图中区别是 export QT_QPA_FB_TSLIB=1 的存在，对于开发板原装支持的屏幕，无论4.3寸还是7 寸屏，都是无需tslib校准的。如果使用另外其他厂商的屏幕，则可能需要tslib校准，而Qt和tslib是需要 声明才能联合使用，详见5.3.1.5.1. Qt和tslib使用建议



```
#! /bin/sh

#you can custom your action after boot in this script
#if you want to run your application after boot

psplash-write "MSG Welcome to Loongson-gd"
psplash-write "PROGRESS 100"
psplash-write "QUIT"
if [ -z "$(cat /proc/cmdline | grep ubifs)" ]; then
    sleep 5
fi
cd /root/logo_player && ./logo_player &

```



```
#! /bin/sh

#you can custom your action after boot in this script
#if you want to run your application after boot

psplash-write "MSG Welcome to Loongson-gd"
psplash-write "PROGRESS 100"
psplash-write "QUIT"
if [ -z "$(cat /proc/cmdline | grep ubifs)" ]; then
    sleep 5
fi
export QT_QPA_FB_TSLIB=1
cd /root/logo_player && ./logo_player &

```

下面是对boot_run.sh的内容进行解析。

psplash-write是关于psplash的命令， psplash是开机启动的进度条的负责程序。当运行至boot_run.sh 的时候，认为系统已经启动完成，所以做出下面的三条指令

```
psplash-write "MSG Welcome to Loongson-gd"
```

这条指令是提示语显示， Welcome to Loongson-gd这句话是System banner，也就是会话终端启动时的 显示语句。于buildroot的BR2_TARGET_GENERIC_ISSUE决定。

```
psplash-write "PROGRESS 100"

psplash-write "QUIT"
```

这两条指令是将**进度条设置为100%**并且**退出进度条的控制**，此后进度条就会停在100%状态和显示上述　的提示语。不会自动消除，除非该显示区域进行新的图形显示。　(在对应有进度条的的终端下使用clear命 令或者显示一个图形程序)

然后是下面这段代码的解析

```
if [ -z "$(cat /proc/cmdline | grep ubifs)" ]; then sleep 5

fi
```

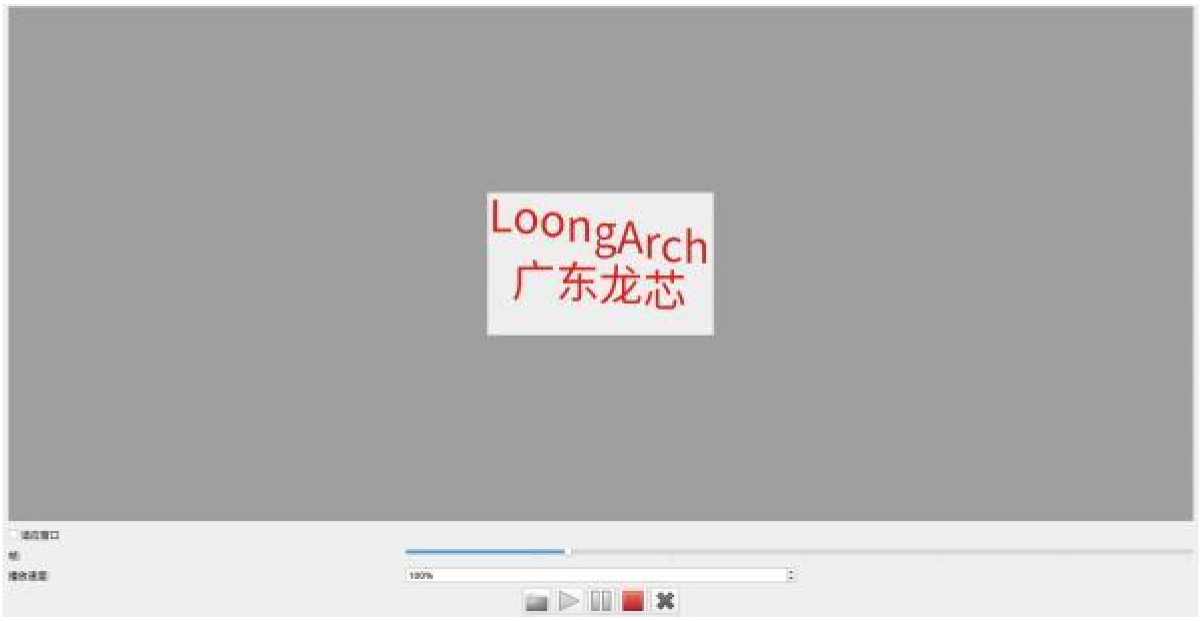
由于执行顺序的问题，系统启动时， boot_run.sh的执行比屏幕终端会话要早，如果Qt程序先于屏幕终　端会话启动，那么会覆盖了Qt程序，问题可以通过睡眠5s来解决。

如果想开机启动其他图形化软件，发现有上述情况，可以参考这个睡眠后再启动的办法。

```
cd /root/logo_player && ./logo_player &
```

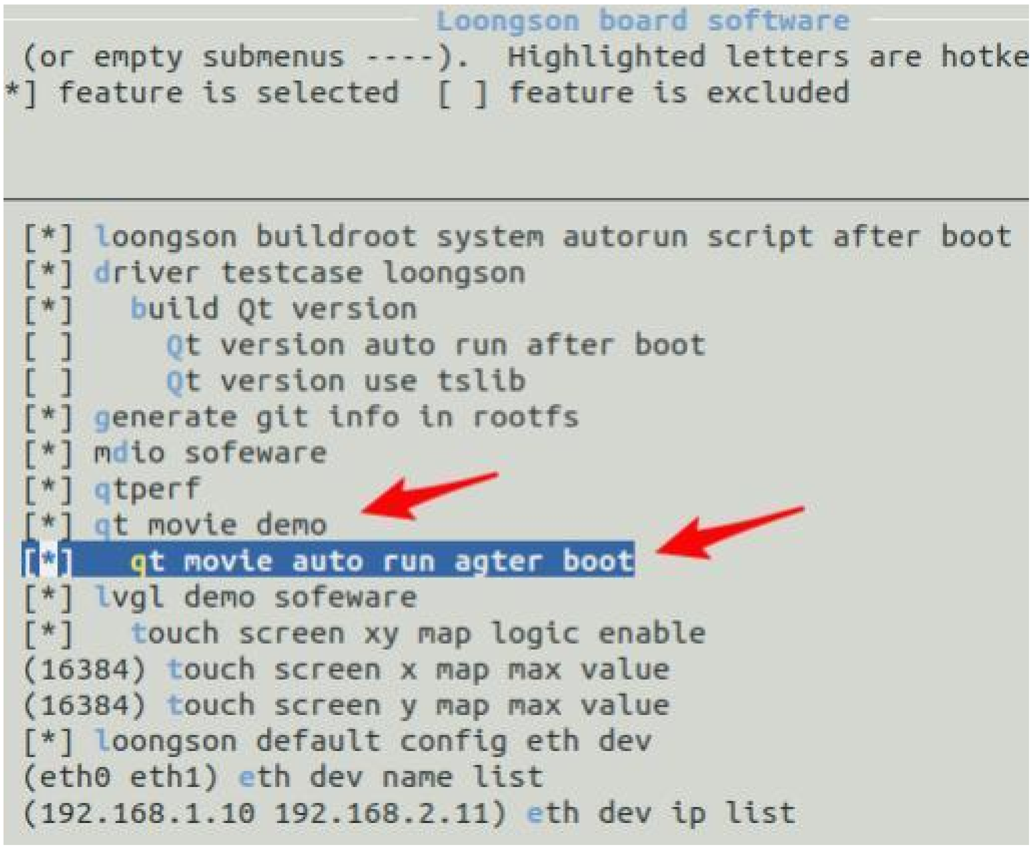
而这一段代码就是运行logo展示程序，由于程序会自动展示运行路径下的logo.gif图片，所以命令中先cd 到程序所在的路径，再运行。是因为logo .gif就在程序所在的文件夹中(/root/logo_player)。

软件运行的**参考**效果图如下：



程序会自动显示一个gif图，字体呈现上下浮动的效果。

这个程序的启动于buildroot的BR2_PACKAGE_QT_MOVIE_AUTOBOOT决定



```
                                  Loongson board software
 (or empty submenus ----).  Highlighted letters are hotke
*] feature is selected  [ ] feature is excluded


 [*] loongson buildroot system autorun script after boot
 [*] driver testcase loongson
 [*]   build Qt version
 [ ]     Qt version auto run after boot
 [ ]     Qt version use tslib
 [*] generate git info in rootfs
 [*] mdio sofeware
 [*] qtperf
 [*] qt movie demo
 [*]   qt movie auto run agter boot
 [*] lvgl demo sofeware
 [*]   touch screen xy map logic enable
 (16384) touch screen x map max value
 (16384) touch screen y map max value
 [*] loongson default config eth dev
 (eth0 eth1) eth dev name list
 (192.168.1.10 192.168.2.11) eth dev ip list
```

5.5.设置RTC时间

请注意， RTC的使用离不开电池的供电，请确保电池供电正常。否则RTC无法读取（会有以下错误 `hwclock: RTC_RD_TIME: Invalid argument`）。

首先说明RTC时间和系统时间。

RTC时间是RTC部件里面保存的时间，这个时间所在的时区认为是UTC。

系统时间是文件系统启动后，系统记录的时间，这个时间的时区会根据/etc/timezone等文件

决定。 date命令可以查看系统的当前时间，以buildroot系统为例，系统中已经设置了时区是

东八区。

那么输入以下的命令之后就会看到 'Thu Nov 24 10:44:11 CST 2022' 这个结果的时间里面有一个CST。

```
date
```

以下命令是设置系统时间的一个例子。

```
date -s '2022-11-24 10:45:00'
```

如果想要改写rtc时间，可以使用 hwclock 命令（需要root权

限）。 以下命令会把系统时间写到rtc中 (系统时间 -> RTC时

间)

```
hwclock -w
```

以下命令会把rtc时间写到系统时间中 （ RTC时间 ->系统时间）

```
hwclock -s
```

但是由于设置了时区，hwclock -w直接执行会把时区的时间当做UTC时间作为RTC时间。从而出现问题。

所以建议写入RTC时间的命令为(-u代表以UTC时间写入)：

```
hwclock -w -u
```

如果板卡上存在多个rtc部件。不指定部件时，默认操作 /dev/rtc0

假设想操作 /dev/rtc1 那么只需要在执行的命令中添加参数：-f /dev/rtc1

5.6.设置网络ip

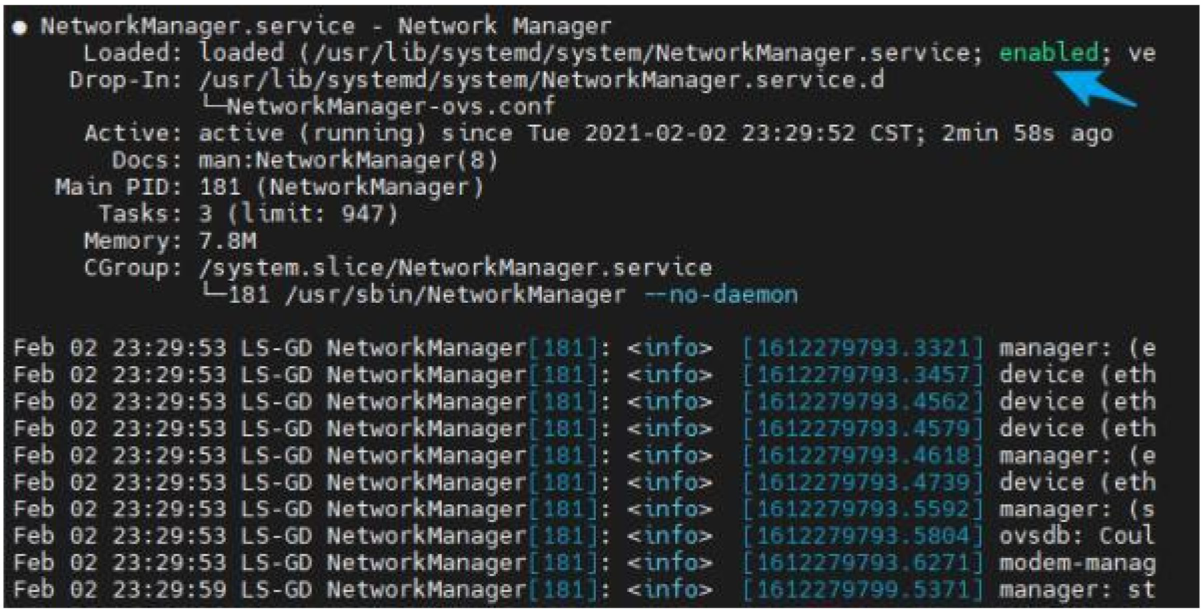
通常可以使用 ifconfig 或者 ip 命令来设定网络地址。

而buildroot构建的busybox系统中，如果安装的是全量系统，则会开机自动设置ip地址。 **eth0的ip设置 为192.168.1.10** ， 如果有网卡2，则对应的设备**eth1的ip设置为192.168.2.11**

以上的自动设置是依靠Network-Manager软件来实现的。如何判断是否使用了Network-Manager软件可 以参考以下方式:

使用如下命令，如果能够返回对应服务的信息，那么就是在使用Network-Manager

```
systemctl status NetworkManager
```



```
● NetworkManager.service - Network Manager
     Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; ve
    Drop-In: /usr/lib/systemd/system/NetworkManager.service.d
             └─NetworkManager-ovs.conf
     Active: active (running) since Tue 2021-02-02 23:29:52 CST; 2min 58s ago
       Docs: man:NetworkManager(8)
   Main PID: 181 (NetworkManager)
      Tasks: 3 (limit: 947)
     Memory: 7.8M
     CGroup: /system.slice/NetworkManager.service
             └─181 /usr/sbin/NetworkManager --no-daemon

Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.3321] manager: (e
Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.3457] device (eth
Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.4562] device (eth
Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.4579] device (eth
Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.4618] manager: (e
Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.4739] device (eth
Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.5592] manager: (s
Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.5804] ovsdb: Coul
Feb 02 23:29:53 LS-GD NetworkManager[181]: <info>  [1612279793.6271] modem-manag
Feb 02 23:29:59 LS-GD NetworkManager[181]: <info>  [1612279799.5371] manager: st
```

默认添加的网络配置如下：

```
# nmcli c show
NAME            UUID                                  TYPE      DEVICE
eth0-connection  1bbd9bed-6870-4098-a4f3-06a5bb11ad3d  ethernet  eth0
[root@LS-GD ~]# cd /etc/NetworkManager/system-connections

[root@LS-GD system-connections]# ls

eth0-connection.nmconnection

[root@LS-GD system-connections]# cat eth0-connection.nmconnection

[connection]

id=eth0-connection
uuid=1bbd9bed-6870-4098-a4f3-

06a5bb11ad3d type=ethernet

interface-name=eth0
```

```
mac-address-blacklist=

 [ipv4]
address1=192.168.1.10/2
4 dns-search=
method=manual

 [ipv6]
addr-gen-mode=stable-privacy
dns-search=
method= auto
```

可能在使用时会发现以下问题：在文件系统中想设置网络ip，比如eth0的ip设置为192.168.3.22。使用了 ifconfig 命令如下：

```
ifconfig eth0 192.168.3.22
```

一开始ip确实是这个，但是后来eth0的ip又变回了192.168.1.10。这是正常的。因为Network-Manager 这个软件会定时检测eth0的ip，如果变了，就会自动设置为默认的ip，也就是192.168.1.10。

如果确实要修改ip地址，应该利用Network-Manager的命令。比如： nmtui和 nmcli。

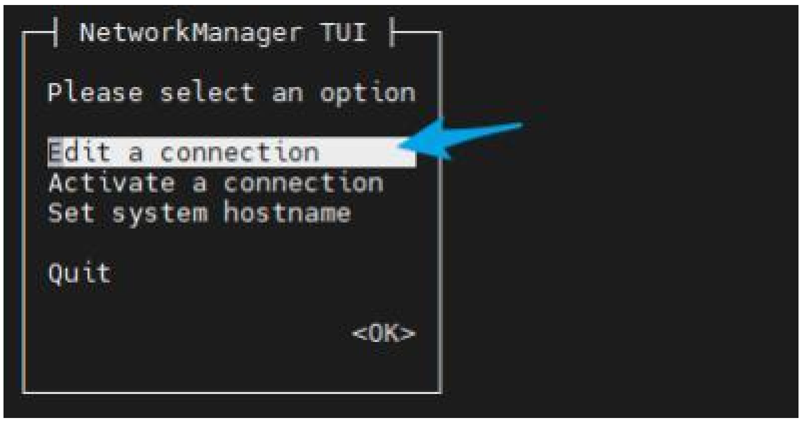
nmcli命令是类似 ifconfig 命令一样，输入参数设置ip的。使用方法不赘述，可以自行搜索。

除了可以运行 nmtui 进行修改，也可以直接编辑 /etc/NetworkManager/system-connections/eth0-connection.nmconnection 中的 [ipv4]-->address1 的 ip，然后运行 nmcli connection reload 和 nmcli connection up eth0-connection 命令使之生效。还可运行 systemclt stop NetworkManager 命令关闭 NetworkManager服务，之后通过 ifconfig 或 ip 命令进行修改。

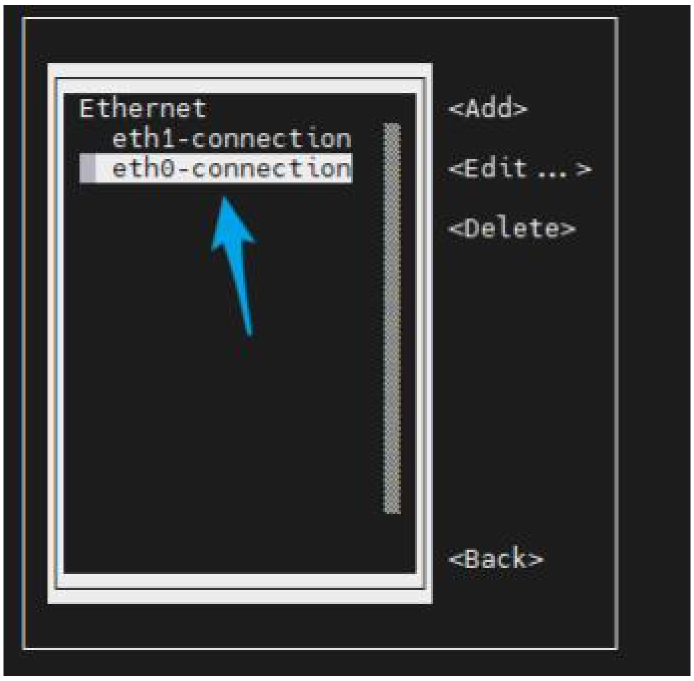
nmtui命令是会提供一个简易图形化设置的窗口。在终端下输入

```
nmtui
```

之后就会弹出一个界面。这个界面的操作就是上下左右移动，然后回车确认。下面是使用2k1000 星云板 作为例子，设置ip，此板卡存在两个网口。

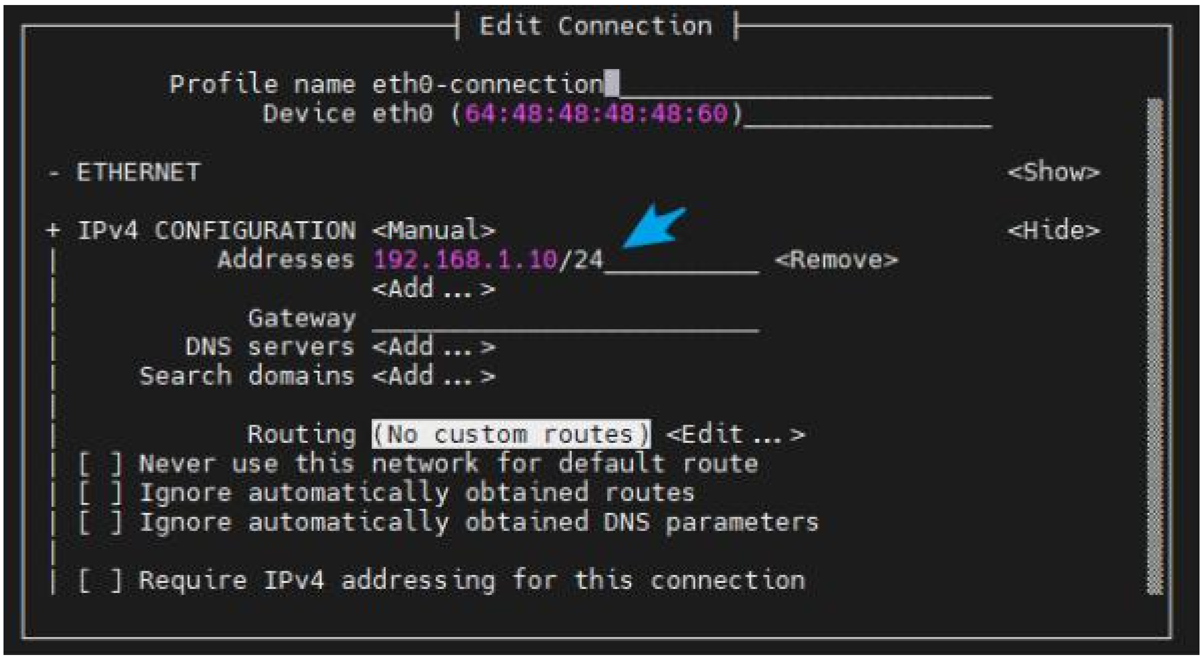


可以看见下面有两个配置选项。这两个选项是在系统第一次开机的时候写入的。但是这个动作是人为规定的。即配置的名字只是一个例子，不是每个设备自动后接"-connection"。
要修改eth0 的 ip 是 192.168.3.22。所以选 "eth0-connection" 那项。

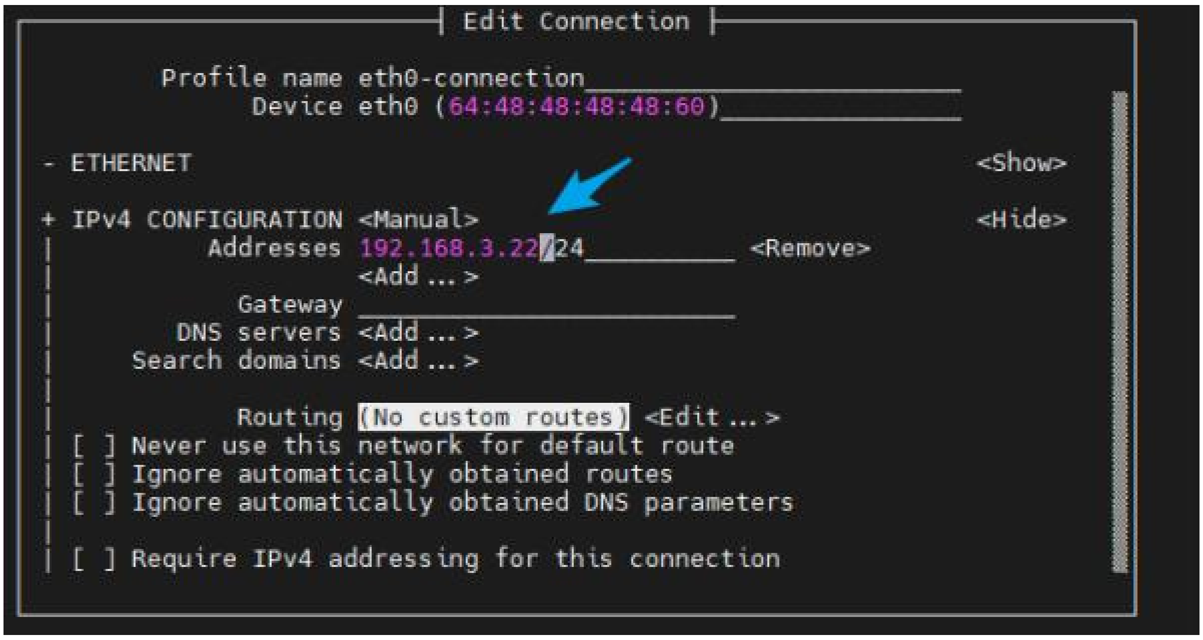


```
Ethernet                 <Add>
  eth1-connection
  eth0-connection        <Edit ... >

                         <Delete>




                         <Back>
```

通过键盘的方向键上下左右移动到可以修改ip的那一栏，修改ip即可，后面的"/24"指的是掩码的长度， 也就是24bit，即255.255.255.0。

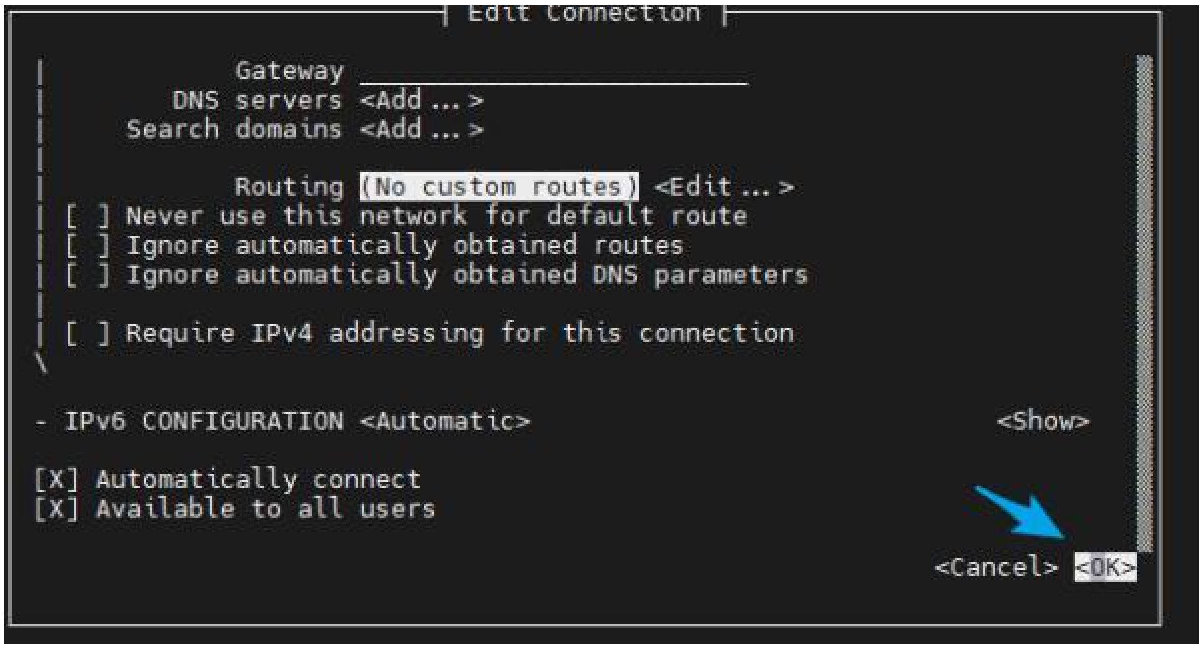


```
                         ┤ Edit Connection ├

        Profile name eth0-connection________________________
              Device eth0 (64:48:48:48:48:60)________________

- ETHERNET                                                    <Show>

+ IPv4 CONFIGURATION <Manual>                                 <Hide>
|          Addresses 192.168.1.10/24__________ <Remove>
|                    <Add ... >
|            Gateway _________________________
|        DNS servers <Add ... >
|     Search domains <Add ... >
|
|            Routing (No custom routes) <Edit ... >
| [ ] Never use this network for default route
| [ ] Ignore automatically obtained routes
| [ ] Ignore automatically obtained DNS parameters
|
| [ ] Require IPv4 addressing for this connection
```



```
                         ┤ Edit Connection ├

        Profile name eth0-connection________________________
              Device eth0 (64:48:48:48:48:60)________________

- ETHERNET                                                    <Show>

+ IPv4 CONFIGURATION <Manual>                                 <Hide>
|          Addresses 192.168.3.22/24__________ <Remove>
|                    <Add ... >
|            Gateway _________________________
|        DNS servers <Add ... >
|     Search domains <Add ... >
|
|            Routing (No custom routes) <Edit ... >
| [ ] Never use this network for default route
| [ ] Ignore automatically obtained routes
| [ ] Ignore automatically obtained DNS parameters
|
| [ ] Require IPv4 addressing for this connection
```

通过键盘的方向键上下左右移动到 "OK" 那一栏，然后回车，确认设置。



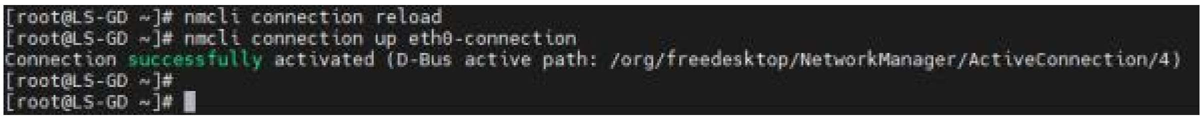
```
┤ Edit Connection ├

|          Gateway ____________________
|      DNS servers <Add ... >
|   Search domains <Add ... >
|
|          Routing (No custom routes) <Edit ... >
| [ ] Never use this network for default route
| [ ] Ignore automatically obtained routes
| [ ] Ignore automatically obtained DNS parameters
|
| [ ] Require IPv4 addressing for this connection
\

- IPv6 CONFIGURATION <Automatic>                    <Show>

[X] Automatically connect
[X] Available to all users

                                              <Cancel> <OK>
```

随后退出软件即可。

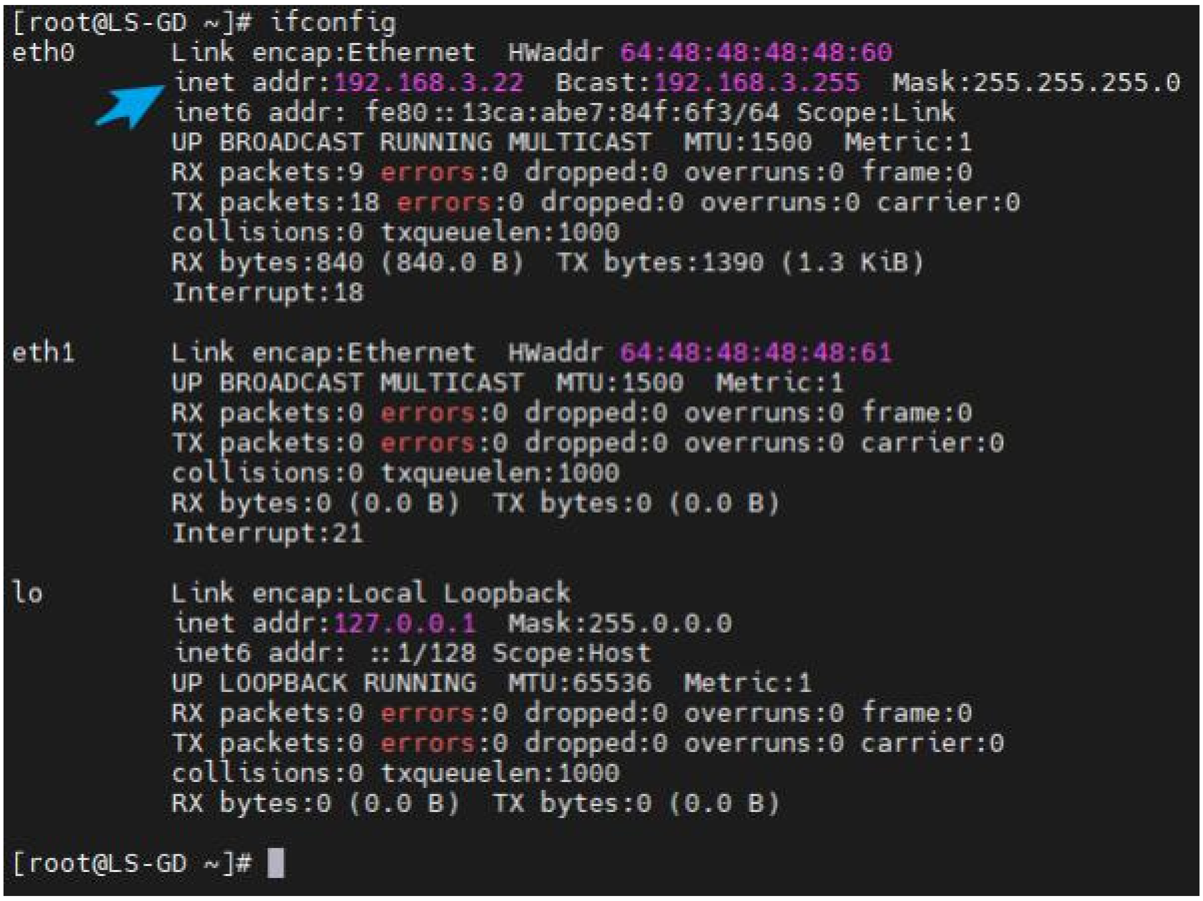
设置完成之后，下次开机就会按照配置设置ip地址。

如果想立即生效配置的ip，可以参考以下命令，假设对应网口的配置名为: eth0-connection

```
nmcli connection reload
nmcli connection up eth0-connection
```



```
[root@LS-GD ~]# nmcli connection reload
[root@LS-GD ~]# nmcli connection up eth0-connection
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/4)
[root@LS-GD ~]#
[root@LS-GD ~]#
```



```
[root@LS-GD ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 64:48:48:48:48:60
          inet addr:192.168.3.22  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::13ca:abe7:84f:6f3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:840 (840.0 B)  TX bytes:1390 (1.3 KiB)
          Interrupt:18

eth1      Link encap:Ethernet  HWaddr 64:48:48:48:48:61
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:21

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[root@LS-GD ~]#
```

5.6.1.NetworkManager的启用和禁用

如果认为Networkmanager的自动设置过于干扰网络设备，想手动或者使用脚本设置网络ip。可以禁用 Networkmanager服务。

开机禁用 Networkmanager 命令参考如下：

```
systemctl disable NetworkManager
```

开机启用 Networkmanager 命令参考如下：

```
systemctl enable NetworkManager
```

马上停止 Networkmanager 命令参考如下：

```
systemctl stop NetworkManager
```

启动 Networkmanager 命令参考如下：

```
systemctl start NetworkManager
```

5.7.WIFI使用

在支持 WIFI 模块驱动、固件的情况下， ifconfig 能看到 wlan 设备：

```
ifconfig -a

...

wlan0     Link encap:Ethernet  HWaddr D6:EF:AB:03:06:69
            BROADCAST MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

5.7.1.NetworkManager配置WIFI

5.7.1.1.nmcli命令行工具配置WIFI

1.扫描 WIFI

```
$ nmcli dev wifi
IN-USE  BSSID              SSID                          MODE   CHAN  RATE
    SIGNAL   BARS  SECURITY
          50:88:11:AE:48:79  wifi-test                       Infra  1     260
Mbit/s  84      ****  WPA2
          ...
```

2.连接WIFI

```
$ nmcli dev wifi connect "wifi-test" password "wifi-test123"
Device 'wlan0' successfully activated with 'cada4448-d4fd-4e2f-a664-
efb9e4932378'.
```

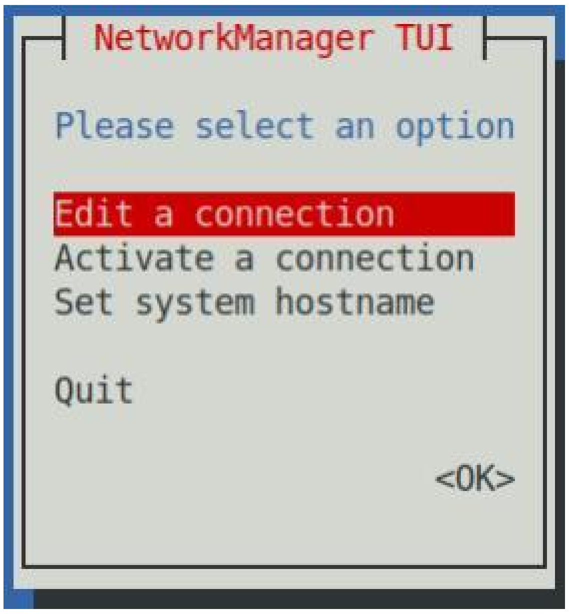
5.7.1.2.nmtui图形化工具配置WIFI

> 建议在 ssh 终端调用 nmtui

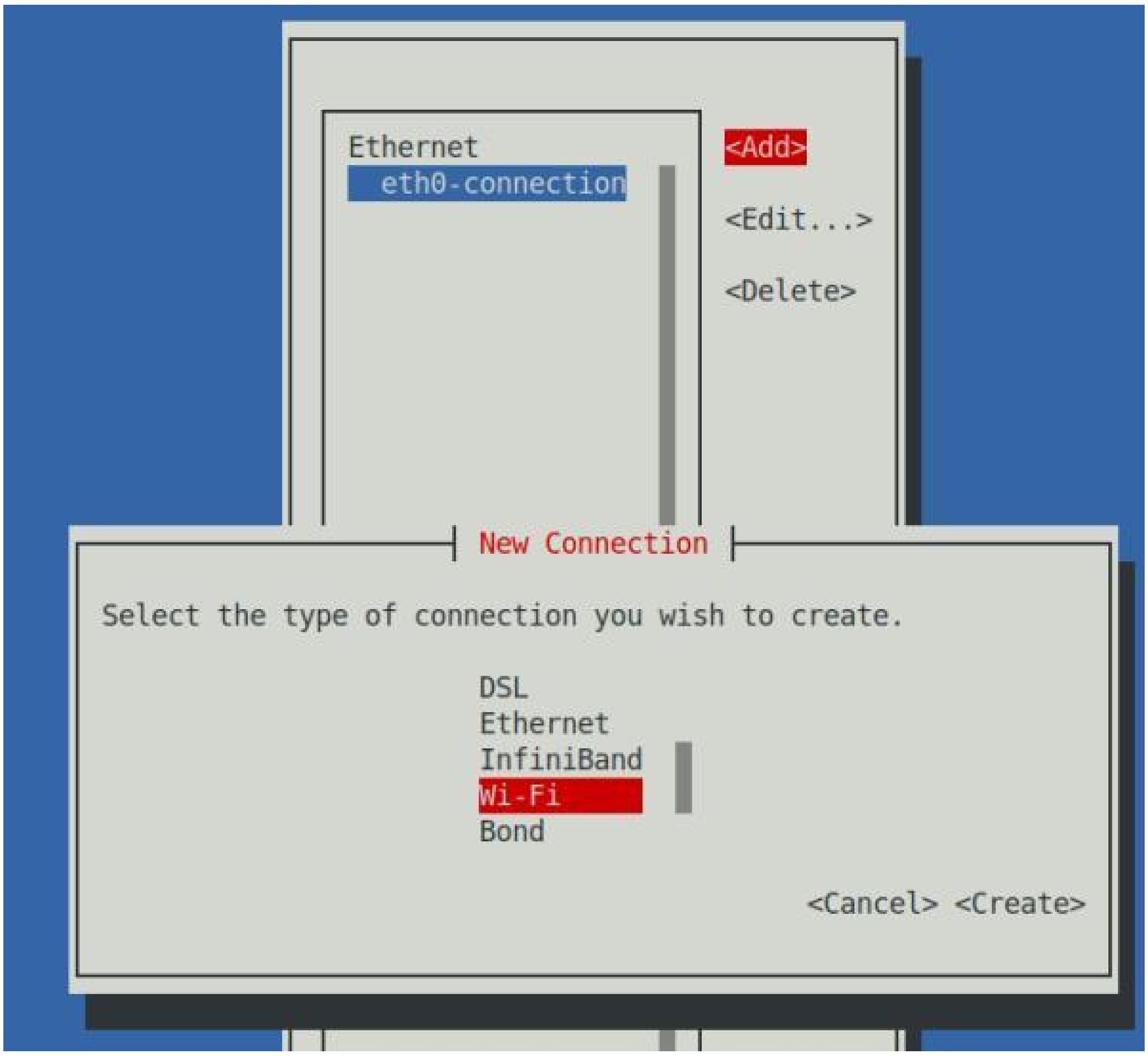
1.调用 nmtui 工具打开界

面　 2.选择" **Edit a**

connection"。

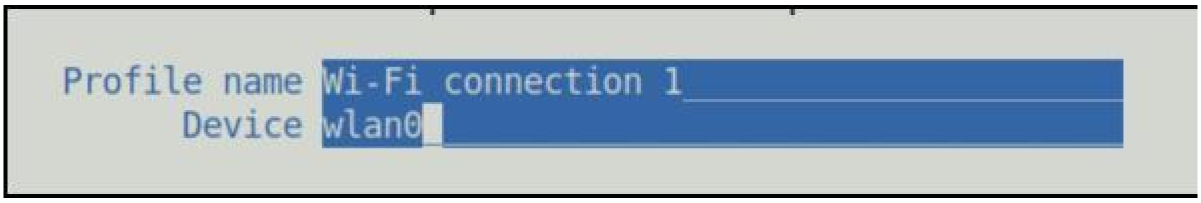

```
NetworkManager TUI

Please select an option

Edit a connection
Activate a connection
Set system hostname

Quit

                  <OK>
```

3.选择右栏添加"**Add**"， 并选中 "**Wi-Fi**"。



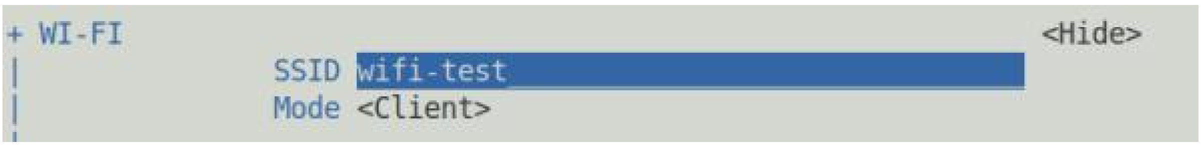
```
Ethernet
  eth0-connection                <Add>

                                 <Edit...>

                                 <Delete>

┤ New Connection ├

Select the type of connection you wish to create.

                 DSL
                 Ethernet
                 InfiniBand
                 Wi-Fi
                 Bond

                                    <Cancel> <Create>
```

4.填写" Device"为无线网卡名称"wlan0"。



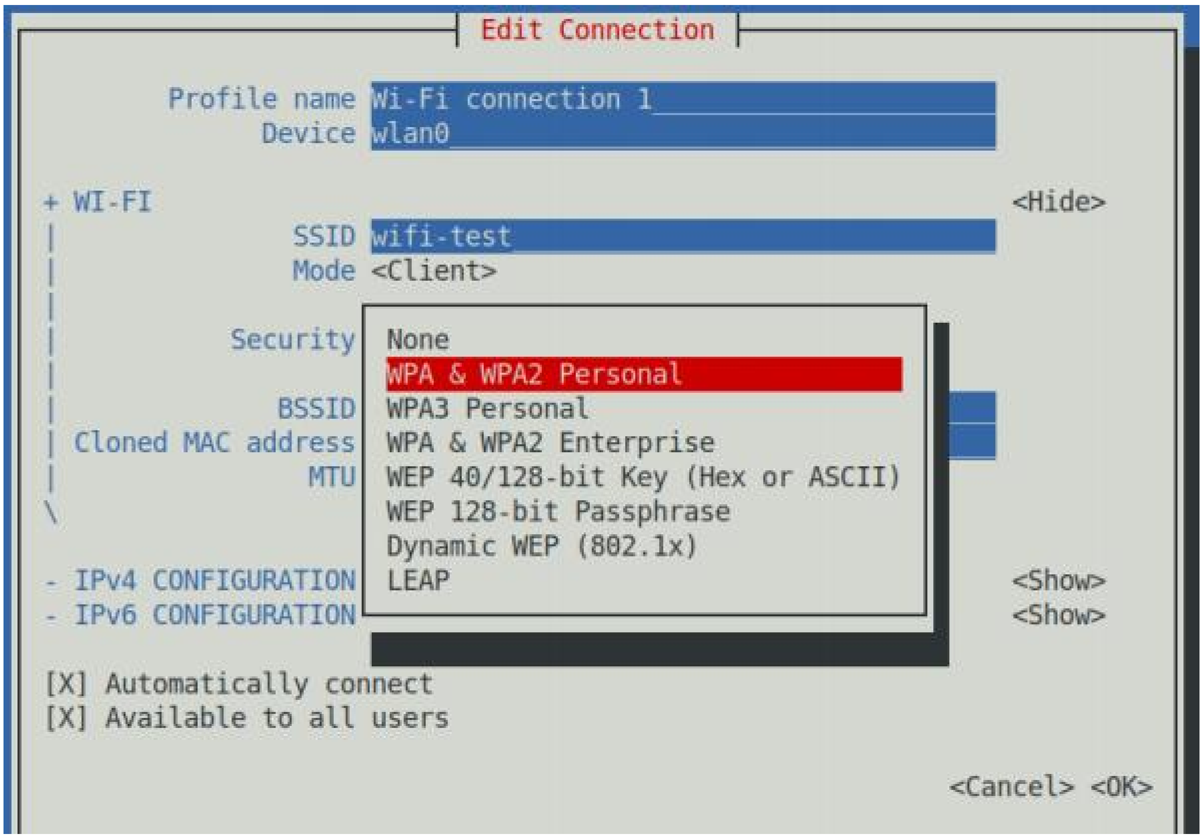
```
Profile name Wi-Fi connection 1
      Device wlan0
```

5.填写"SSID"为要接入的无线网络"wifi-test"。



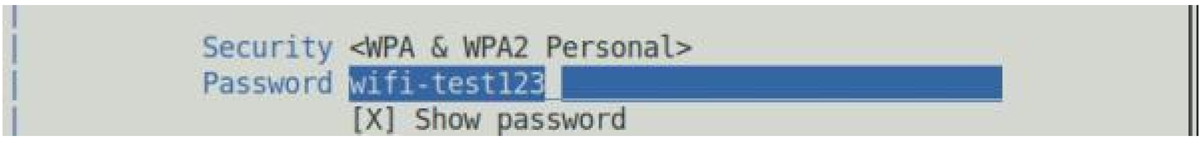
```
+ WI-FI                                                        <Hide>
|             SSID wifi-test
|             Mode <Client>
```

6.选中"Security"， 改为无线网络所使用的加密方式，通常为 "WPA & WPA2 Personal"。



```
Edit Connection

      Profile name Wi-Fi connection 1
            Device wlan0

+ WI-FI                                                      <Hide>
|             SSID wifi-test
|             Mode <Client>
|
|         Security  None
|                   WPA & WPA2 Personal
|            BSSID  WPA3 Personal
| Cloned MAC address WPA & WPA2 Enterprise
|              MTU  WEP 40/128-bit Key (Hex or ASCII)
\                   WEP 128-bit Passphrase
                    Dynamic WEP (802.1x)
- IPv4 CONFIGURATION LEAP                                   <Show>
- IPv6 CONFIGURATION                                        <Show>

[X] Automatically connect
[X] Available to all users

                                                  <Cancel> <OK>
```

7.填写Wifi密码" **Password**"。



```
|           Security <WPA & WPA2 Personal>
|           Password wifi-test123
|                    [X] Show password
```

5.7.2.iw与wpa工具配置WIFI

> 注意： NetworkManager会抢占WIFI控制，因此建议在没有**NetworkManager**运行的系统下使 用wpa工具

1.iw 工具扫描 **WIFI SSID**

```
$ iw dev wlan0 scan | grep SSID
          SSID: wifi-test
          ...
```

2.**wpa_passphrase**设置 **WIFI**密码

```
$ wpa_passphrase "wifi-test" "wifi-test123" > /etc/wpa_supplicant.config
```

3.**wpa_supplicant**连接 **WIFI**

```
$ wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.config
Successfully initialized wpa_supplicant
```

4.**dhcpcd** 协商 IP：

```
$ dhcpcd wlan0

DUID 00:01:00:01:27:ac:2c:96:00:0e:02:00:21:db

wlan0: IAID 02:00:21:db
```

```
wlan0: soliciting a DHCP lease
wlan0: soliciting an IPv6 router
wlan0: offered 192.168.10.252 from 192.168.10.1
wlan0: probing address 192.168.10.252/24
wlan0: leased 192.168.10.252 for 43200 seconds
wlan0: adding route to 192.168.10.0/24
wlan0: adding default route via 192.168.10.1
forked to background, child pid 240

$ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:0E:02:00:21:DB
          inet addr:192.168.10.252  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20e:2ff:fe00:21db/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

5.7.3.WIFI使用问题答疑

5.7.3.1.使用NetworkManager无法连接WIFI

排查思路如下：

1.先关闭 NetworkManager

```
$ systemctl stop NetworkManager
```

2.使用 wpa 工具连接 WIFI

```
$ wpa_passphrase "wifi-test" "wifi-test123" > /etc/wpa_supplicant.config $ wpa_sup
plicant -B -i wlan0 -c /etc/wpa_supplicant.config
$ dhcpcd wlan0
```

3.如果 wpa 工具连接 WIFI 成功，那么有可能是 WIFI芯片不支持 Random Mac 导致的，我们就要禁用 NetworkManager 中的 Random Mac，重启后再使用 NetworkManager 连接

```
$ echo -e "[device]\nwifi.scan-rand-mac-address=no" | tee
/etc/NetworkManager/conf.d/disable-random-mac.conf
$ reboot
```

如果以上思路无法解决问题请反馈。

5.7.3.2.使用传统的 iw 工具无法连接 WIFI

iw 工具只支持 WEP 加密，现在常见的 WIFI 都是 WPA 加密，建议使用 wpa 工具或 NetworkManager
连接。

5.8. DOCKER使用

首先确保内核支持docker，可以用下面的脚本对内核的 defconfig 进行检测
https://docs.docker.com/engine/install/troubleshoot/
https://github.com/coreos/docker/blob/master/contrib/check-config.sh

接下来在 loongnix系统上安装 docker

```
sudo apt install docker-ce
```

如果遇到docker服务因为iptable启动失败的情况，可进行以下操作

```
sudo update-alternatives --set iptables /usr/sbin/iptables-legacy
sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy sudo
systemctl start docker
```

然后就可以在龙芯镜像仓库(https://cr.loongnix.cn/)上搜索拉取镜像

```
docker pull cr.loongnix.cn/library/debian:buster-slim
```

运行容器

```
docker run -d -it --name a1 cr.loongnix.cn/library/debian:buster-slim
```

在容器中运行命令

```
docker exe -it a1 /bin/bash
```

5.9. NFS使用

首先搭建NFS服务器（以x86或龙芯电脑作为服务端为例）

```
sudo apt install nfs-kernel-server
```

启动NFS服务

```
sudo systemctl enable --now nfs-server
```

创建文件夹

```
sudo mkdir -p /media/nfs
```

打开 /etc/exports，写入内容，将NFS路径设置为刚才建立的文件夹

```
/media/nfs            192.168.1.0/24(rw,sync,no_subtree_check)
```

exportfs

```
sudo exportfs -arv
```

如果允许客户端写入文件，要加上权限

```
sudo chmod go+w /media/nfs
```

板卡连接NFS服务器则很简单（192.168.1.2是NFS服务器IP）

```
mount -t nfs4 192.168.1.2:/media/nfs /media/share
```

5.10.音频使用

5.10.1.ALSA工具使用

列出声卡设备

```
arecord -l
```

播放音频

```
aplay test.wav
```

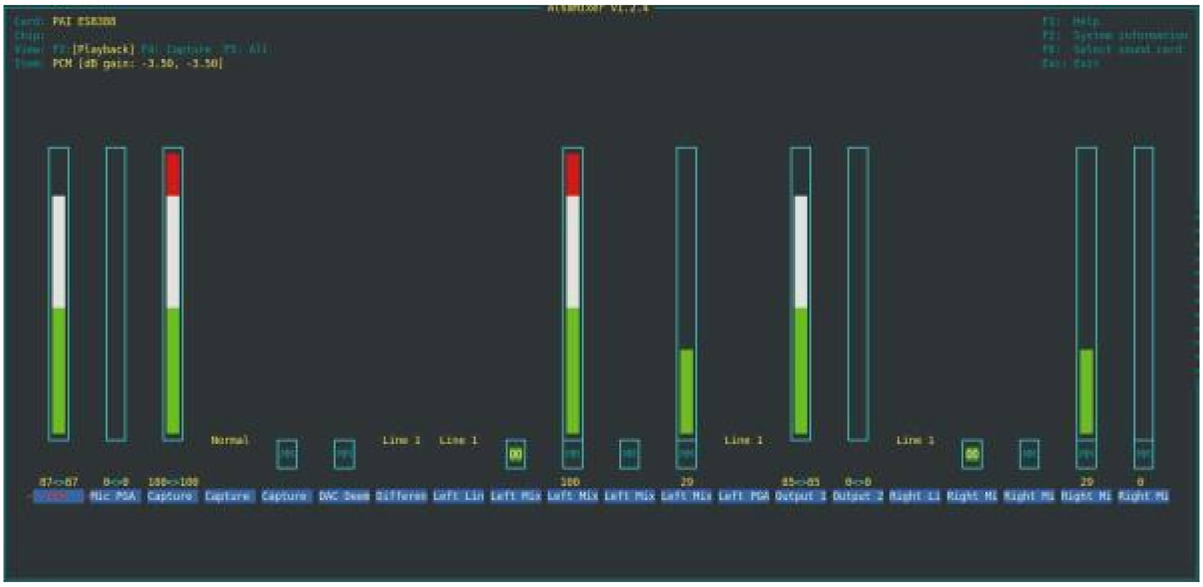
录音

```
arecord -d 10 -r 48000 -c 1 -f S16_LE audio.wav
```

alsamixer 调整音量

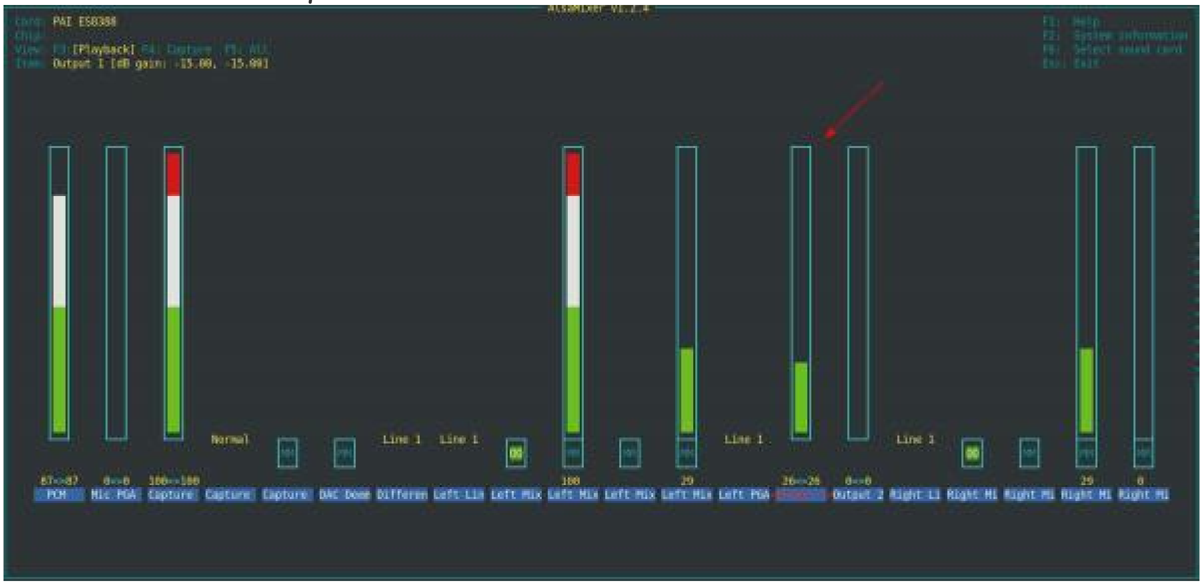
建议使用ssh登录后运行如下命令

```
alsamixer
```



```
Card: PAI ES8388                                                     F1:  Help
Chip:                                                                F2:  System information
View: F3:[Playback] F4: Capture  F5: All                             F6:  Select sound card
Item: PCM [dB gain: -3.50, -3.50]                                    Esc: Exit
```

按**左右键**将光标移到 ***Output1*** 上，再按**上下调整音量**



```
Card: PAI ES8388
Chip:
View: F3:[Playback] F4: Capture  F5: All
Item: Output 1 [dB gain: -15.00, -15.00]
```

保存 **alsamixer** 参数

```
alsactl store 0
```

5.10.2.音频使用答疑

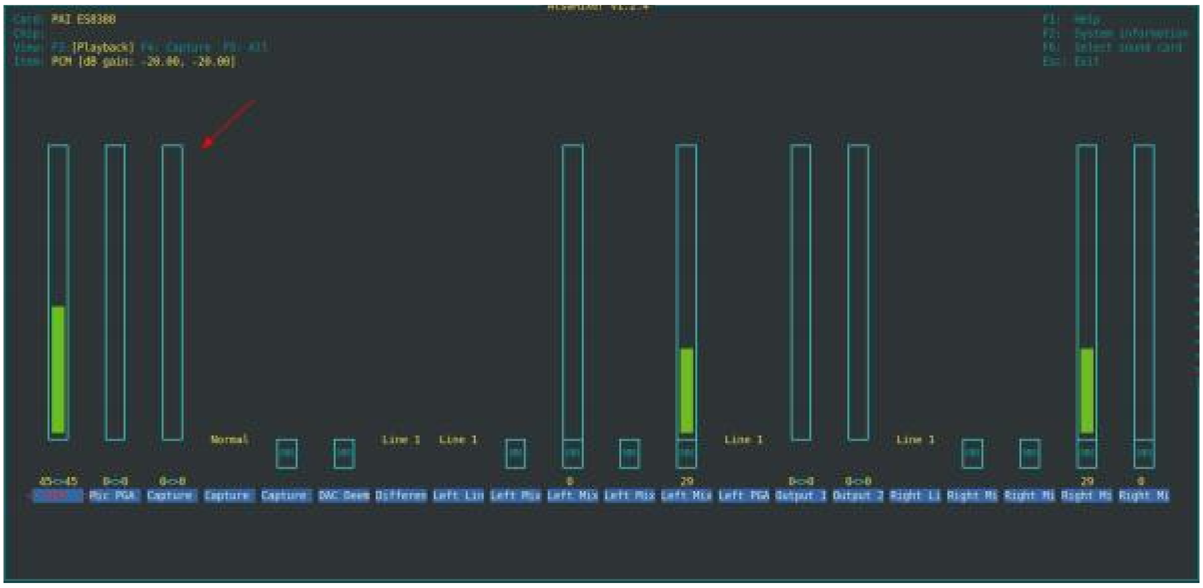
5.10.2.1.alsactl提示 asound.state lock error

这是由于没有 /var/lock 目录引起的，手动创建一个 /var/lock 目录即可

```
mkdir /var/lock
alsactl store 0
```

5.10.2.2.安装新的系统后播放或录音出现问题

1.检查 alsamixer 中显示的参数和 /var/lib/alsa/asound.state 中参数是否一致



```
Card: PAI ES8388
Chip:
View: F3:[Playback] F4: Capture  F5: All
Item: PCM [dB gain: -20.00, -20.00]
```

```
$ cat /var/lib/alsa/asound.state

state.ES8388 {
        control.1 {
```

```
                    name 'Capture Digital Volume'
                    value.0  192
                    value.1  192
                    comment {
                            access 'read write'
                            type
                            INTEGER
                            count 2
                            range '0 - 192'
                            dbmin -
                            9600 dbmax
```

可以看到**Capture**项参数不对，**asound.state**中**value**已经给到**192**，而**alsamixer**中柱状条显示**0**　2.参数不一致说明系统中默认配置没生效，检查是否存在 /var/lock　目录

```
$ ls /var/lock
ls: cannot access '/var/lock': No such file or directory
```

3.手动创建一个 **/var/lock**　目录

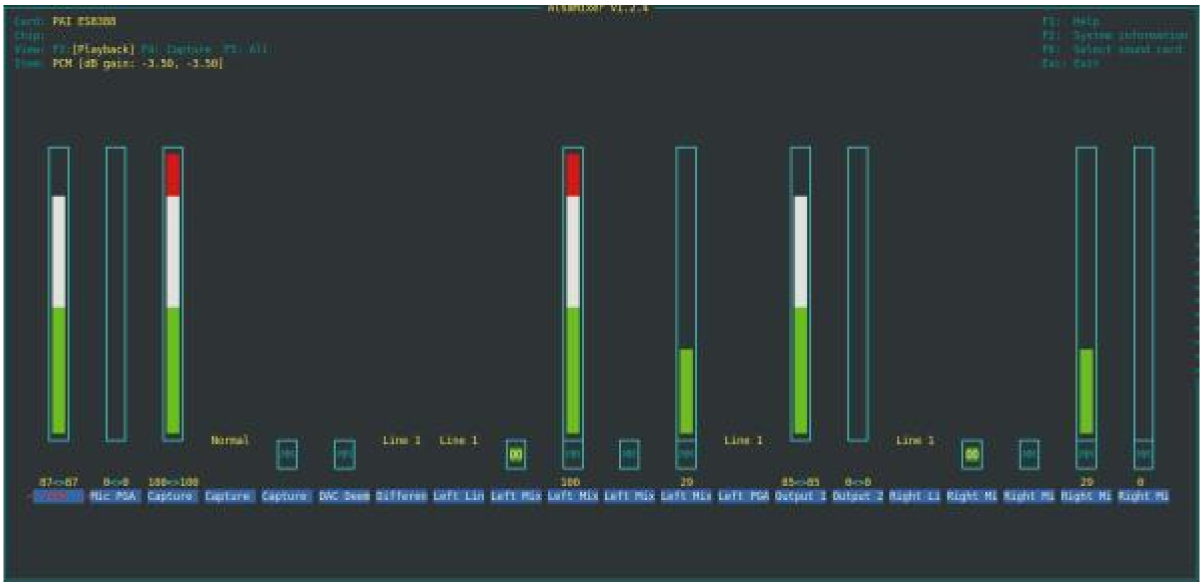
```
mkdir /var/lock
```

4.使用系统默认参数

```
alsactl restore 0 -f /var/lib/alsa/asound.state
```

5.此时再查看 alsamixer 发现参数统一

```
alsamixer
```



```
Card: PAI ES8388
Chip:
View: F3:[Playback] F4: Capture  F5: All
Item: PCM [dB gain: -3.50, -3.50]
```

5.11.无线网卡作为AP（热点）使用

首先先要确保所用的无线网卡，比如USB形式的无线网卡，插入板卡后，板卡可以创建一个网卡设备。 最简单的方法就是可以作为接收wifi使用，参考在[5.7.WIFI使用](#)。

本节将无线网卡作为AP，即创建热点。这样可以保证板卡的IP地址固定。更加便于连接ssh来调试（因为 IP地址固定）。

本节用到的工具有 dhcpd 和 hostapd 。

- hostapd的作用是提供AP连接
- dhcpd的作用是可以为连接热点的设备分配IP，否则热点是连不上的

下面演示例子的相关信息如下，这些信息都是为了演示或者实际情况下得出的，可根据自身环境进行不同的修改：

- 演示用的系统是buildroot
- hostapd的热点配置文件是/root/hostapd.conf
- dhcpd的配置文件是/etc/dhcp/dhcpd.conf
- 板卡IP为192.168.3.1
- 板卡的无线网卡设备名为wlan0
- 热点名字为 testap，密码为12345678
- 板卡中dhcpd服务分配的IP地址为 192.168.3.2至192.168.3.50

5.11.1.hostapd的热点配置

首先创建hostapd所需的热点配置文件，参考如下：

```
interface=wlan0
driver=nl80211
ssid=testap
hw_mode=g
channel=10
macaddr_acl=0
auth_algs=3
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
rsn_pairwise=TKIP CCMP
```

其中“wpa=2”这个字段及其之后的字段，如果不设置，那么这个热点就没有密码限制，直接可以连接。 “ssid”和“wpa_passphrase”字段就是热点的名字和密码，其他字段根据实际情况可自行更改。

5.11.2.dhcpd的分配规则

接下来是/etc/dhcp/dhcpd.conf。直接在该文件后添加以下字段

```
subnet 192.168.3.0 netmask 255.255.255.0

{

     range 192.168.3.2 192.168.3.50;

     option routers 192.168.3.1;

     option domain-name-servers 8.8.8.8; }
```

5.11.3.启动AP

启动AP的流程为：

1. hostapd运行
2. 网卡设备IP设置
3. dhcpd运行

创建一个脚本， 一键进行启动，如果文件为apcreate.sh（名字是随便起的）。

```
#! /bin/sh

hostapd_conf="/root/hostapd.conf"
wlan_dev="wlan0"
wlan_ip="192.168.3.1"

NAME="dhcpd"
DAEMON="/usr/sbin/${NAME}"
CFG_FILE="/etc/default/${NAME}"

# Read configuration variable file if it is present
[ -r "${CFG_FILE}" ] && . "${CFG_FILE}"

start_ap() {
        hostapd $hostapd_conf -B
        ifconfig $wlan_dev $wlan_ip

        # start dhcpd
        test -d /var/lib/dhcp/ || mkdir -p /var/lib/dhcp/
        test -f /var/lib/dhcp/dhcpd.leases || touch /var/lib/dhcp/dhcpd.leases start-stop-
        daemon -S -q -x ${DAEMON} -- -q $OPTIONS $INTERFACES
}

stop_ap() {
        killall hostapd 2>/dev/null
        start-stop-daemon -K -q -x ${DAEMON}
}

restart_ap() {
```

```
tip() {
      echo "tip:"
      echo "$1 (mean restart setup ap)" echo
      "$1 (mean restart setup ap)" echo "$1
      (mean restart setup ap)" echo ""
      echo "$1 start (mean start setup ap)" echo
      "$1 s (mean start setup ap)"
      echo "$1 stop (mean close ap)"
      echo "$1 close (mean close ap)" echo
      "$1 e (mean close ap)"
      echo "$1 restart (mean restart ap setup)" echo
      "$1 reload (mean restart ap setup)"  echo "$1 r
      (mean restart ap setup)"
}

if [ $# -eq 0 ]; then
      restart_ap
      exit 0
fi

   case "$1" in
         start|s)
            start_ap
            ;;
      stop|end|close)
            stop_ap
```

需要注意的是，当你想设置网段不是192.168.3.xxx时，请同时修改该文件，即wlan0的ip 和/etc/dhcp/dhcpd.conf中指定的网段要一致。
记得为此文件添加可执行权限，参考指令：

```
chmod a+x apcreate.sh
```

上述脚本中包括三个功能

- 开启热点
- 关闭热点
- 重启热点

开启热点的参考使用指令为(下面指令是指任意一条都行，都会执行开始热点的动作)：

```
./apcreate.sh start
./apcreate.sh s
```

关闭热点的参考使用指令为(下面指令是指任意一条都行，都会执行关闭热点的动作)：

```
./apcreate.sh stop
./apcreate.sh end
./apcreate.sh close
```

重启热点的参考使用指令为(下面指令是指任意一条都行，都会执行重启热点的动作)：

```
./apcreate.sh
./apcreate.sh restart
./apcreate.sh reload
./apcreate.sh r
```

需要留意的是，开启热点的动作不能在热点还在的时候多次执行，比如下面的指令一起执行了，会导致热点无法用：

```
./apcreate.sh start
./apcreate.sh start
```

重启热点的动作也建议不要多次调用。如果是使用ssh连接了，重启热点会导致一段时间的卡顿，之后才 会恢复。

5.12.关于dhcpd服务

本节只是说明buildroot文件系统默认没有开机自启dhcpd服务的原因

因为buildroot做出来的文件系统，默认会对网卡做ip设置，然而dhcpd启动的时候会对所有ip进行检 查，如果没有ip对应的网段的dhcpd声明，那么就会返回错误。

```
[FAILED] Failed to start DHCP server.
```

如果开机自启dhcpd服务，那么就会一直出现dhcpd的错误，那么会影响使用。所以buildroot那边默认 是没开systemd的dhcpd服务

但如果想自行开启dhcpd自启，那么可以参考下面的步

骤 创建 dhcpd.service 文件（没vim命令的话用vi

）

```
vim /usr/lib/systemd/system/dhcpd.service
```

文件内容如下：

```
 [Unit]
Description=DHCP server
After=network.target

 [Service]
Type=forking
StartLimitIntervalSec=0
Restart=always
RestartSec=2
```

```
KillSignal=SIGINT
EnvironmentFile=-/etc/default/dhcpd

 [Install]
WantedBy=multi-user.target
```

确定编写好之后，使能开机启动的命令参考如下：

```
systemctl enable dhcpd.service
```

确定编写好之后，如果想马上启动该服务，命令参考如下：

```
systemctl start dhcpd.service
```

如果想让dhcpd服务不重试，可以把dhcpd.service文件中的 Restart 和 RestartSec 字段删除，参考如下

```
 [Unit]
Description=DHCP server
After=network.target

 [Service]
Type=forking
StartLimitIntervalSec=0
PIDFile=/run/dhcpd.pid
ExecStart=/usr/sbin/dhcpd -q -pf /run/dhcpd.pid $OPTIONS $INTERFACES
KillSignal=SIGINT
EnvironmentFile=-/etc/default/dhcpd

 [Install]
WantedBy=multi-user.target
```

5.13.网口MAC地址修改

关于网口的MAC地址修改方式，下面提供2种参考修改方式， 一种是在文件系统内修改， 一种是在uboot （固件）中修改。

5.13.1.文件系统中修改网口MAC地址

下面提供的参考方法是使用ifconfig来修改。但此种修改方式只是本次系统运行时修改有效，下次重启则 还是按照uboot中存放的mac地址为准。

假设要修改eth0这个网卡的mac地址，想修改为

56:11:22:33:44:55 地址。 修改前 eth0 信息如下：

```
[root@LS-GD ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:A9:2F:C7:67:66
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:18 Base address:0x8000
```

参考命令如下：

```
ifconfig eth0 down
ifconfig eth0 hw ether 56:11:22:33:44:55
```

修改后 eth0 信息如下：

```
[root@LS-GD ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 56:11:22:33:44:55
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:18 Base address:0x8000
```

可见修改成功。

5.13.2.uboot中修改网口MAC地址

对于嵌入式板卡， uboot里面有环境变量存放mac地址。

要想通过uboot修改，首先进入uboot的命令行操作模式，就是开机的时候通过调试串口， 一直按着'c'， 则会进入到uboot的命令行模式。

```
U-Boot 2022.04-v2.0.0-00265-g34430548 (Nov 21 2023 - 09:33:23 +0800)

CPU:   LA264
Speed: Cpu @ 997 MHz/ Mem @ 800 MHz/ Bus @ 200
MHz Model: loongson-2k300
Board: LS2K300-MINI-
DP DRAM:  512 MiB
512 MiB

Jump to board_init_r....
Core:  35 devices, 19 uclasses, devicetree: board
SF: Detected w25q64cv with page size 256 Bytes, erase size 4 KiB, total 8 MiB bdinfo is
in spi-flash
cam_disable:1, vpu_disable:1, pcie0_enable:0, pcie1_enable:1
Loading Environment from SPIFlash... OK
Cannot get ddc bus
In:    serial gpiobtn
Out:    serial vidconsole
Err:   serial vidconsole
Net:   eth0: ethernet@40040000, eth1: ethernet@40050000
************************** Notice ************************** Press c to
enter u-boot console, m to enter boot menu
************************************************************ Autoboot
in 0 seconds

【此处是刷屏了，正常来说只会看到 => 】


=>
```

输入命令，就能看到一系列的uboot的环境变量

```
print
```

就会发现这样的一个变量，这个就是网卡的mac地址

。 ethaddr是eth0

```
ethaddr=52:a9:2f:c7:67:66
```

假设要修改 eth0 为

52:a9:2f:c7:67:99 那么参考命

令为：

```
setenv ethaddr '52:a9:2f:c7:67:99'

saveenv
```

随后重启(reboot命令)，进入系统，就会看见mac地址修改成功。

```
[root@LS-GD ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:A9:2F:C7:67:99
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:18 Base address:0x8000
```

5.14.ADC使用

2K300 集成了1路12bit 8通道的ADC。其电压量程是 -1.8v~1.8V，使用时将电压发生器的地与开发板的 地引脚相连，电压输出端与 ADC 引脚相连，查看检测到的值（这里测量ADC0电压为例）。

```
# cat /sys/bus/iio/devices/iio\:device0/in_voltage0_raw
1351
```

得到的数据做以下转换得到电压。 2^12 为4096（即0~4095）故计算公式为将测量得到的值除以4095 再 剩以最大量程1.8v。

以本示例来说，电压大约是 0.59V

```
1351/4095 x 1.8v ≈ 0.59v
```

5.15. LCD 背光控制

1.关闭背光

```
# echo 0 > /sys/devices/platform/backlight/backlight/backlight/brightness
```

2.打开背光

```
# echo 1 > /sys/devices/platform/backlight/backlight/backlight/brightness
```

5.16 USB摄像头

将USB 摄像头接到板卡上。

1.查看USB 摄像头设备节点名称

```
# ls /dev/video*
/dev/video0  /dev/video1
```

2.控制USB摄像头拍照，并查看照片信息

```
# v4l2grab -d /dev/video0 -o 1.jpg #
file 1.jpg
1.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length
16, baseline, precision 8, 640x480, components 3
```

3.使用mjpg-streamer播放usb 摄像头的视频流并本地观看（摄像头需支持mjpg 流）

```
# mjpg_streamer -i "input_uvc.so -d /dev/video0" -o "output_viewer.
MJPG Streamer Version: git rev: 45bf210ba715ca304f48e9ebd60235f48e5fe6ca
 i: Using V4L2 device.: /dev/video0
 i: Desired Resolution: 640 x 480
 i: Frames Per Second.: -1
 i: Format............: JPEG
 i: TV-Norm...........: DEFAULT
UVCIOC_CTRL_ADD - Error at Pan (relative): Inappropriate ioctl for device (25)
UVCIOC_CTRL_ADD - Error at Tilt (relative): Inappropriate ioctl for device (25)
UVCIOC_CTRL_ADD - Error at Pan Reset: Inappropriate ioctl for device (25)
UVCIOC_CTRL_ADD - Error at Tilt Reset: Inappropriate ioctl for device (25)
UVCIOC_CTRL_ADD - Error at Pan/tilt Reset: Inappropriate ioctl for device (25)
UVCIOC_CTRL_ADD - Error at Focus (absolute): Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at Pan (relative): Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at Tilt (relative): Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at Pan Reset: Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at Tilt Reset: Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at Pan/tilt Reset: Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at Focus (absolute): Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at LED1 Mode: Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at LED1 Frequency: Inappropriate ioctl for device (25)
UVCIOC_CTRL_MAP - Error at Disable video processing: Inappropriate ioctl for
device (25)
UVCIOC_CTRL_MAP - Error at Raw bits per pixel: Inappropriate ioctl for device (25)
```

```
  o: input plugin.....: 0: input_uvc.so
```

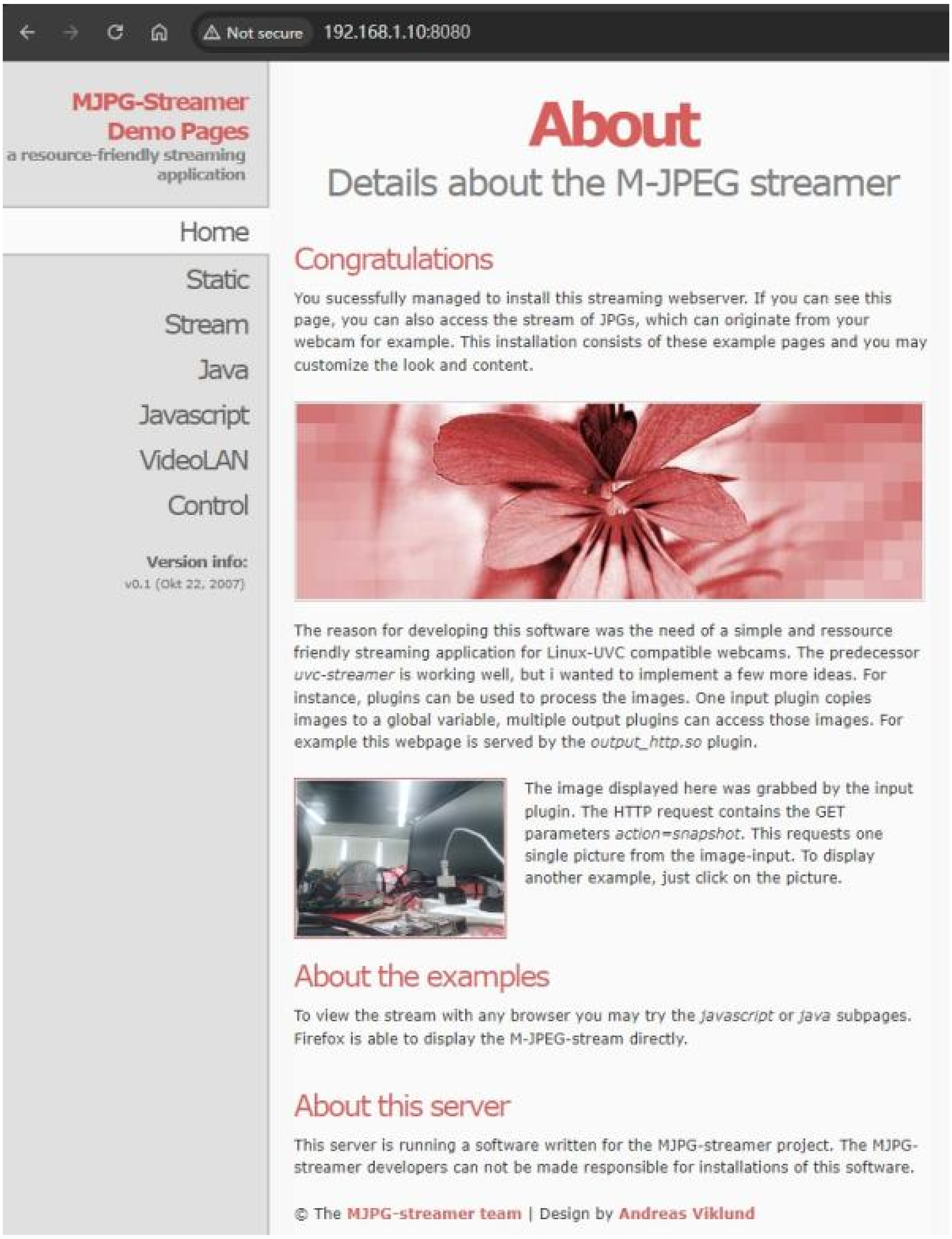
在lcd 屏可以实时观看视频流

4. 使用mjpg-streamer 播放usb 摄像头的视频流并通过浏览器查看（摄像头需支持mjpg 流）

```
# mjpg_streamer -i "input_uvc.so -d /dev/video0" -o "output_http.so -w
/usr/share/mjpg-streamer/www"
```

```
MJPG Streamer Version: git rev: 45bf210ba715ca304f48e9ebd60235f48e5fe6ca

 i: Using V4L2 device.: /dev/video0 i:

 Desired Resolution: 640 x 480

 i: Frames Per Second.: -1
 i: Format............: JPEG
 i: TV-Norm...........: DEFAULT
UVCIOC_CTRL_ADD - Error at Pan (relative): Inappropriate ioctl for device (25)

UVCIOC_CTRL_ADD - Error at Tilt (relative): Inappropriate ioctl for device (25)

UVCIOC_CTRL_ADD - Error at  Pan Reset: Inappropriate ioctl for device (25)

UVCIOC_CTRL_ADD - Error at  Tilt Reset: Inappropriate ioctl for device (25)

UVCIOC_CTRL_ADD - Error at  Pan/tilt Reset: Inappropriate ioctl for device (25)

UVCIOC_CTRL_ADD - Error at  Focus (absolute): Inappropriate ioctl for device (25)

UVCIOC_CTRL_MAP  -  Error   Pan (relative): Inappropriate ioctl for device (25)

at UVCIOC_CTRL_MAP - Error  Tilt (relative): Inappropriate ioctl for device (25)

at UVCIOC_CTRL_MAP - Error  Pan Reset: Inappropriate ioctl for device (25)

at UVCIOC_CTRL_MAP - Error  Tilt Reset: Inappropriate ioctl for device (25)

at UVCIOC_CTRL_MAP - Error  Pan/tilt Reset: Inappropriate ioctl for device (25)

at UVCIOC_CTRL_MAP - Error  Focus (absolute): Inappropriate ioctl for device (25)

at UVCIOC_CTRL_MAP - Error  LED1 Mode: Inappropriate ioctl for device (25)

at UVCIOC_CTRL_MAP - Error  LED1 Frequency: Inappropriate ioctl for device (25)

at UVCIOC_CTRL_MAP - Error  Disable video processing: Inappropriate ioctl for

at UVCIOC_CTRL_MAP - Error  Raw bits per pixel: Inappropriate ioctl for device

at device (25)
                            /usr/share/mjpg-streamer/www/
UVCIOC_CTRL_MAP - Error

at (25)                     8080

        o: www-folder-path......: (null)

          o: HTTP TCP port........: disabled

 o: HTTP Listen Address..: o:   enabled

 username:password....:

               o: commands............:
```

在PC端打开浏览器输入http://ip:8080 即可，ip 是板卡的IP地址，我这里是192.168.1.10.



About

Details about the M-JPEG streamer

Congratulations

You sucessfully managed to install this streaming webserver. If you can see this page, you can also access the stream of JPGs, which can originate from your webcam for example. This installation consists of these example pages and you may customize the look and content.

The reason for developing this software was the need of a simple and ressource friendly streaming application for Linux-UVC compatible webcams. The predecessor *uvc-streamer* is working well, but i wanted to implement a few more ideas. For instance, plugins can be used to process the images. One input plugin copies images to a global variable, multiple output plugins can access those images. For example this webpage is served by the *output_http.so* plugin.

The image displayed here was grabbed by the input plugin. The HTTP request contains the GET parameters *action=snapshot*. This requests one single picture from the image-input. To display another example, just click on the picture.

About the examples

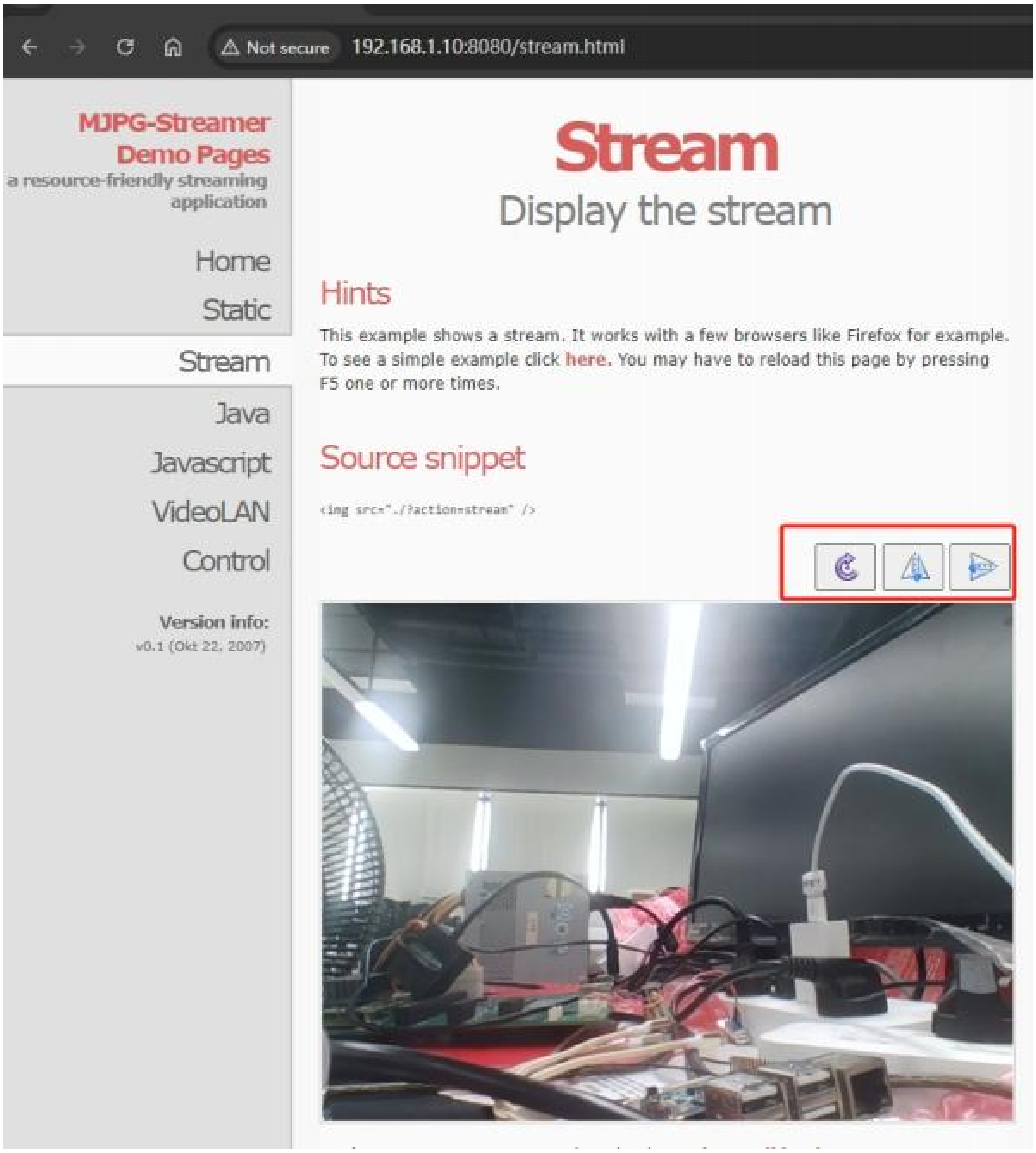
To view the stream with any browser you may try the *javascript* or *java* subpages. Firefox is able to display the M-JPEG-stream directly.

About this server

This server is running a software written for the MJPG-streamer project. The MJPG-streamer developers can not be made responsible for installations of this software.

© The MJPG-streamer team | Design by Andreas Viklund

点击Stream可查看实时视频流，并且支持对图像进行水平和垂直方向的翻转操作。


Not secure 192.168.1.10:8080/stream.html

MJPG-Streamer Demo Pages
a resource-friendly streaming application

Home
Static
Stream
Java
Javascript
VideoLAN
Control

Version info:
v0.1 (Okt 22, 2007)

Stream
Display the stream

Hints

This example shows a stream. It works with a few browsers like Firefox for example. To see a simple example click here. You may have to reload this page by pressing F5 one or more times.

Source snippet

```
<img src="./?action=stream" />
```

5.17 MPV播放视频

在 /root 文件夹下，存在 mp4_sample_video 文件夹。

```
[root@LS-GD ~]# cd mp4_sample_video/
[root@LS-GD mp4_sample_video]# ls
SampleVideo_360x240_1mb.mp4
[root@LS-GD mp4_sample_video]# pwd
/root/mp4_sample_video
[root@LS-GD mp4_sample_video]#
```

这是一个测试视频，用于测试mpv是否能使用

。 参考命令为:

```
mpv /root/mp4_sample_video/SampleVideo_360x240_1mb.mp4 --vo=drm
```

使用 mpv 请加入 --vo=drm 参数。

六、开发板功能测试

> 注意：对于部分接口测试（比如can），需要更新对应设备树才能使用，参考 *10.4.设备树选择*

6.1.开发板接口测试

开发板已预置测试用例，在 /root/loongson_test_case下，主要用到 gpio_test、uart_test、usb_test、can_test、can_rate_test、spi_test、pwm_test、rtc_test、lcd_test。

```
# ls /root/loongson_test_case/
  can_rate_test
  can_test
  driver_testcase
  gpio_test
  lcd_test
  pwm_test
  rtc_test
  spi_test
  uart_test
  usb_test
#
```

6.1.1.网口测试

2K300先锋派有一个网口，在示例中我们将板卡与192.168.1.2的服务
器相连： 1.开启网卡

```
# ifconfig eth0 192.168.1.5
```

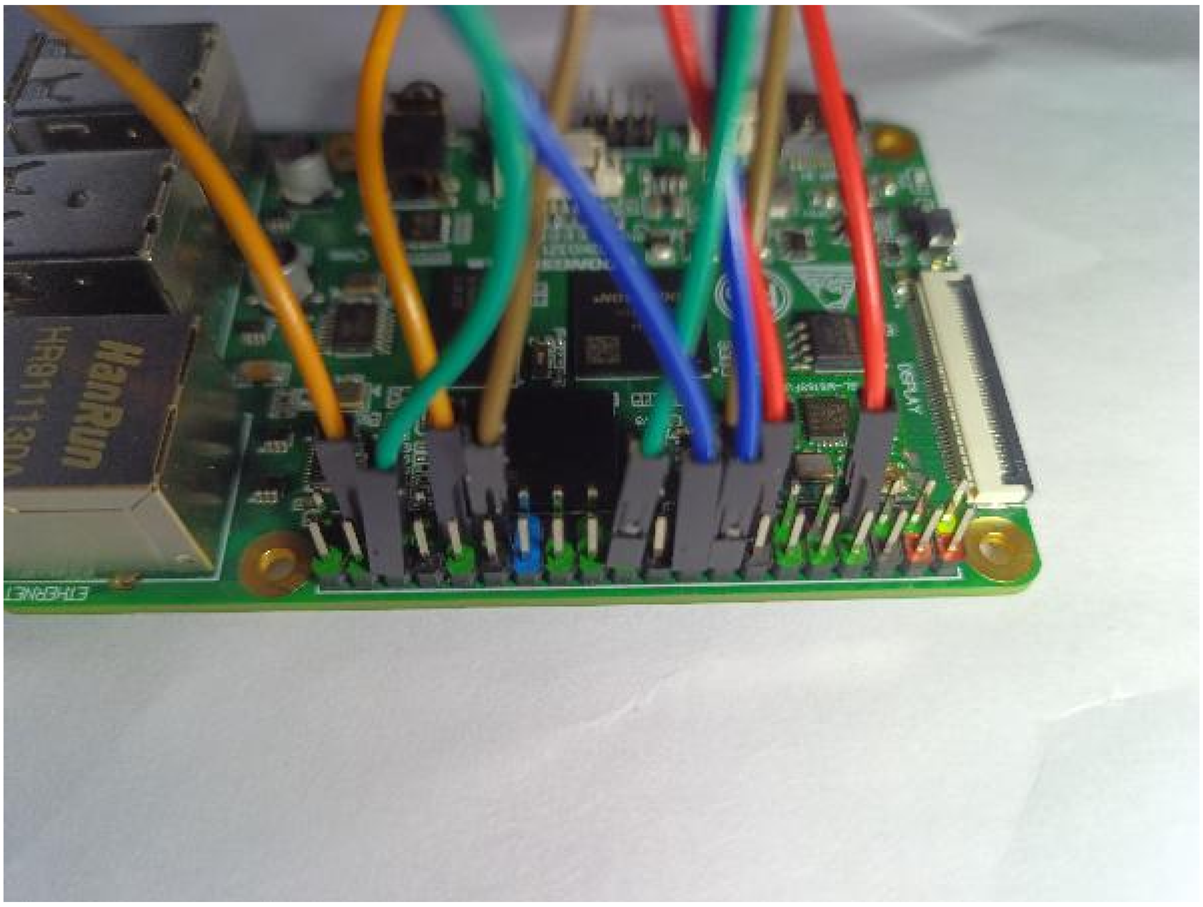
2.Ping 服务器

```
# ping 192.168.1.2 -w 10
PING 192.168.1.2 (192.168.1.2): 56 data     bytes
64 bytes from 192.168.1.2: seq=0 ttl=64     time=2.179 ms
64 bytes from 192.168.1.2: seq=1 ttl=64     time=1.998 ms
64 bytes from 192.168.1.2: seq=2 ttl=64     time=2.091 ms
64 bytes from 192.168.1.2: seq=3 ttl=64     time=1.378 ms
64 bytes from 192.168.1.2: seq=4 ttl=64     time=2.075 ms
64 bytes from 192.168.1.2: seq=5 ttl=64     time=2.070 ms
64 bytes from 192.168.1.2: seq=6 ttl=64     time=2.005 ms
64 bytes from 192.168.1.2: seq=7 ttl=64     time=2.098 ms
64 bytes from 192.168.1.2: seq=8 ttl=64     time=2.152 ms
64 bytes from 192.168.1.2: seq=9 ttl=64     time=2.128 ms

--- 192.168.1.2 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss round-
```

```
 trip min/avg/max = 1.378/2.017/2.179 ms

#
```

6.1.2.GPIO测试

1.将开发板上的 GPIO 接口按下图连接配对连接：J8插座上 7-13相连、15-29相连、31-37相连、16-18相 连、22-36相连



2.进入 gpio_test 目录，运行用例

```
# cd /root/loongson_test_case/gpio_test #

./gpio_test 0 88 69

group_pin(out:pin_88,in:pin_69) test passed! #

./gpio_test 0 81 72

group_pin(out:pin_81,in:pin_72) test passed! #

./gpio_test 0 73 74

group_pin(out:pin_73,in:pin_74) test passed! #

./gpio_test 0 70 71

group_pin(out:pin_70,in:pin_71) test passed! #

./gpio_test 0 84 75

group_pin(out:pin_84,in:pin_75) test passed!
```

6.1.3. UART测试

1.将开发板上的 UART TX-RX 接口按下图连接配对连接：J8插座上 8-10相连

2.进入 uart_test 目录，运行用例

自收发测试，即 rx 与 tx 对接，下面以uart2 为例

```
# cd /root/loongson_test_case/uart_test
# ./uart_test -a /dev/ttyS2
/dev/ttyS2 send: hello,world
/dev/ttyS2 recv: hello,world
uart test passed!
#
```

两个uart 对发测试，即 uart-5 的rx 接 uart-9的tx ，uart-5 的tx 接 uart-9的rx 。

```
# cd /root/loongson_test_case/uart_test
# ./uart_test -a /dev/ttyS5 -b /dev/ttyS9
/dev/ttyS5    send:    hello,world
/dev/ttyS9    recv:    hello,world
uart test passed!
/dev/ttyS9    send:    hello,world
/dev/ttyS5    recv:    hello,world
uart test passed!
```

6.1.4. USB测试

1.插入 FAT32 格式U盘

2.进入 usb_test 目录，运行用例

```
# cd /root/loongson_test_case/usb_test #
./test_usb_RW.sh /dev/sdb1
```

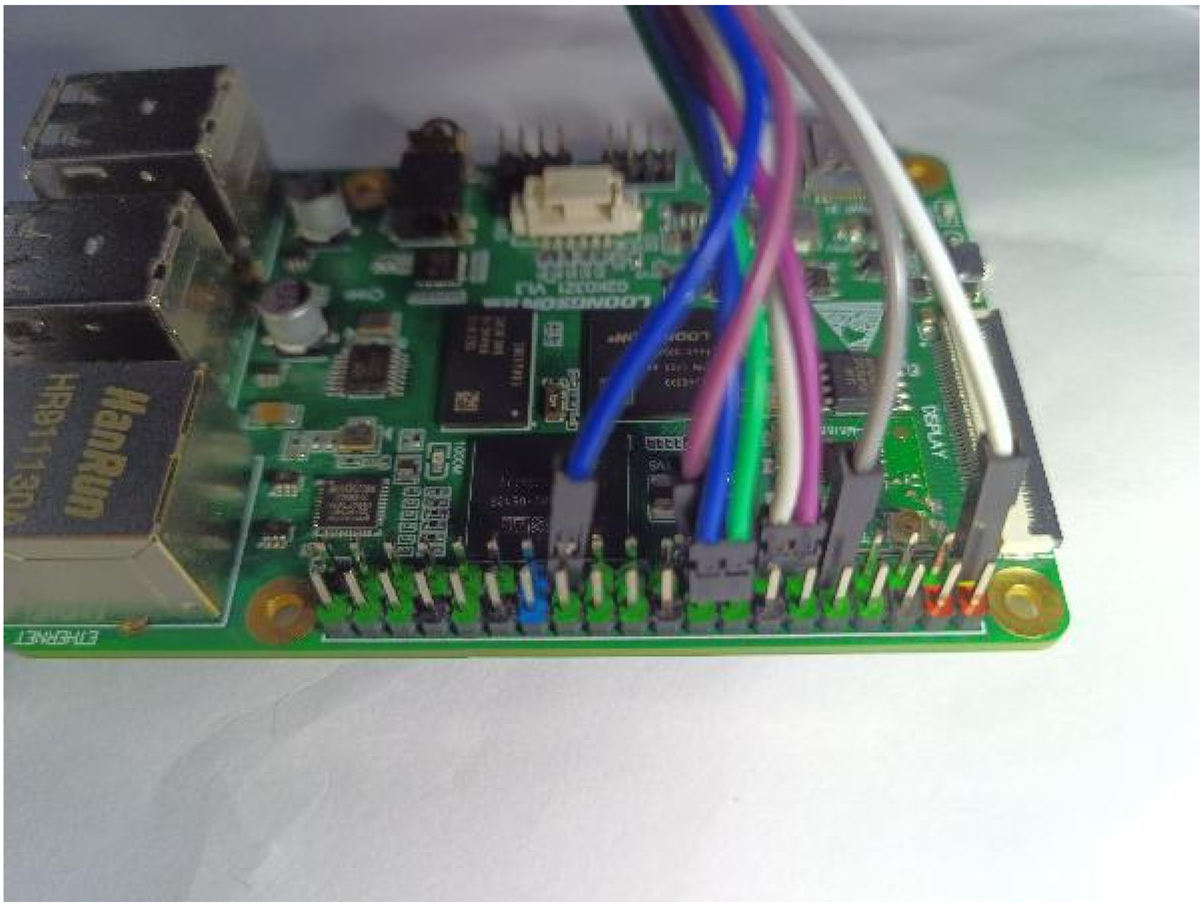
```
probe your rootfs in nand
usb test logic:
  copy a big file from usb to usb
  copy many small file from usb to usb so
speed maybe slow

create a big file

copy big file
1 files (512.0 MiB) copied in 27.5 seconds ( 18.6 MiB/s). delete
big file and md5file

create small file
process: 512 / 512
copy small file
```

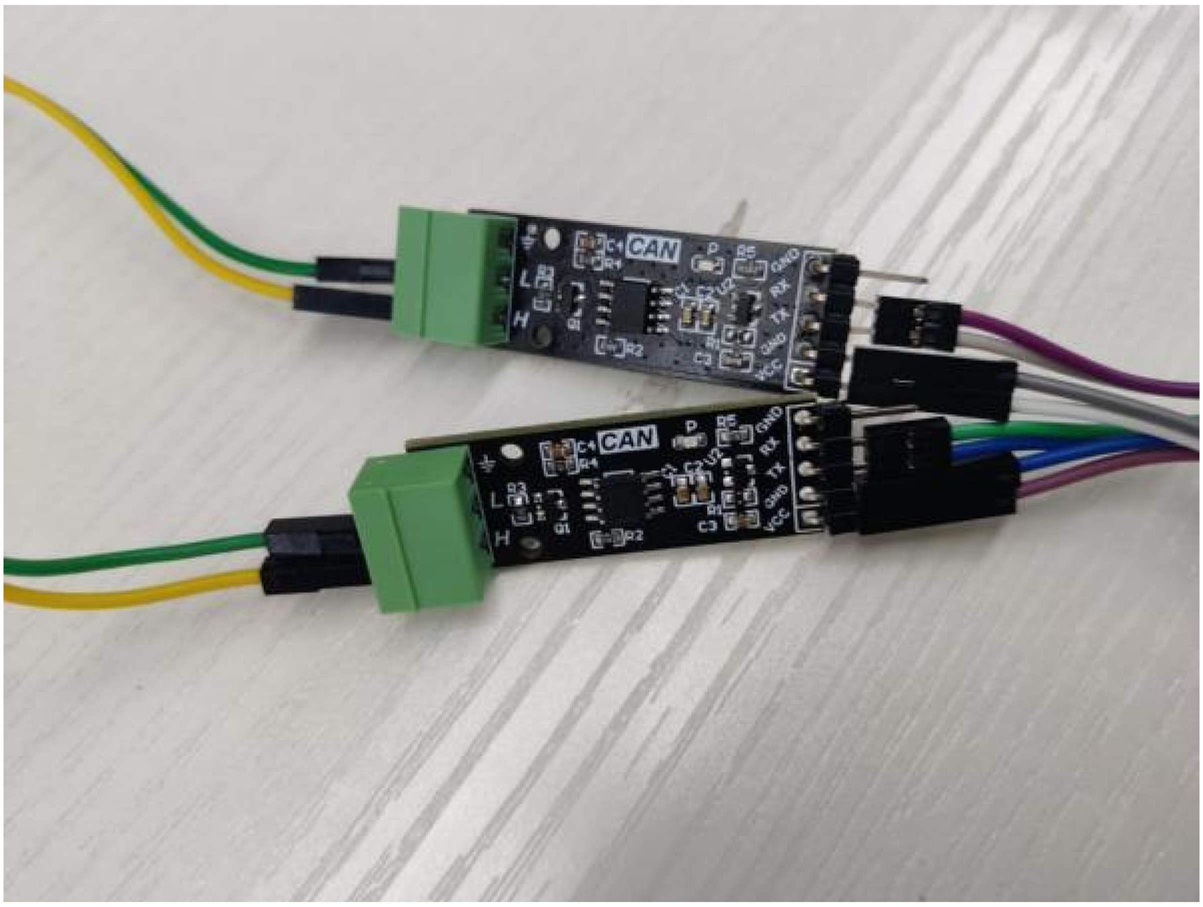
3.插入 *ext4* 格式U盘，重复步骤2

6.1.5.CAN测试

> 测试CAN之前一定要将板卡的复用关系修改

1.准备两个CAN收发器模块，将开发板上的 CAN 接口按下图连接配对连接：收发器TX对板卡CAN-TX、 收发器RX对板卡CAN-RX；两个收发器H-H相连、 L-L相连；收发器接板卡3.3V电。





2.CAN 收发测试，进入 can_test 目录，运行脚本

```
# cd /root/loongson_test_case/can_test #
./can_group -m -a can0 -b can1
can_name:can0, can_id:0x123  rate:250kpbs
cur test baudrate: 250kbps
process: 1 / 1wait recv msg...
can_name:can1, can_id:0x123  rate:250kpbs
cur test baudrate: 250kbps
recevied data: cantest9
test data: cantest9
can test passed!
wait recv msg...
recevied data: cantest9
test data: cantest9
can test passed!
```

6.1.6.SPI测试

该用例是通过读写spi-flash进行测试，只有在flash可读写情况下通过。 1.进入 spi_test 目录，运行用例。

```
# cd /root/loongson_test_case/spi_test #
./spi_test
rx_buf[0] to rx_buf[63]:
ae f1 ff 63 4c 01 00 1c 8c 11 db 28 4d 01 00 1c
ad 91 d5 28 80 01 80 29 8c 11 c0 02 8d f9 ff 63
[illegible]
```

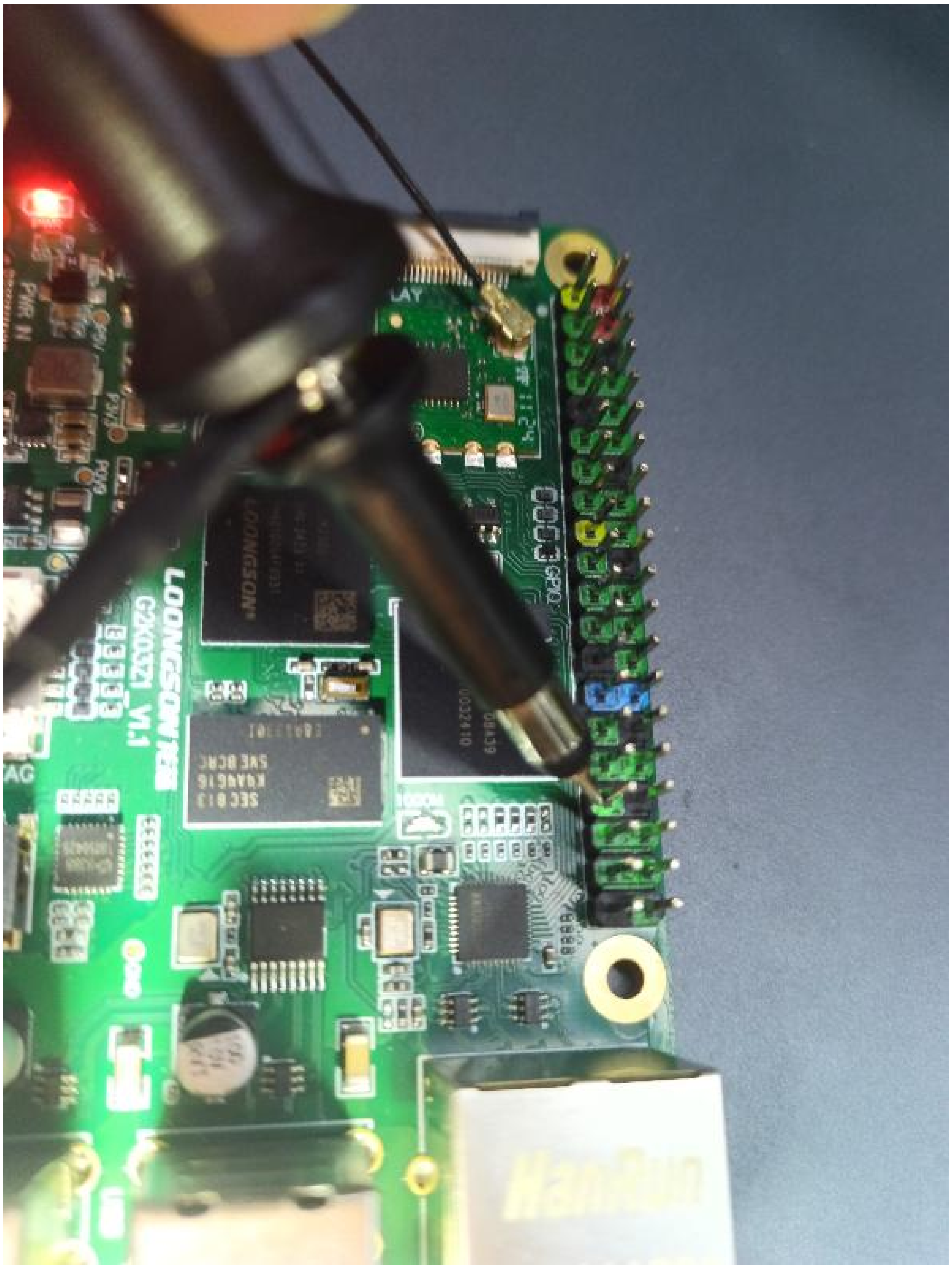
```
rx_buf[0] to rx_buf[63]:
ae f1 ff 63 4c 55 66 77 88 11 db 28 4d 01 00 1c
ad 91 d5 28 80 01 80 29 8c 11 c0 02 8d f9 ff 63
4c 01 00 1c 8c e1 cf 28 80 01 00 4c 04 08 20 15
04 00 00 16 84 00 24 03 01 03 15 00 20 00 00
4c
```

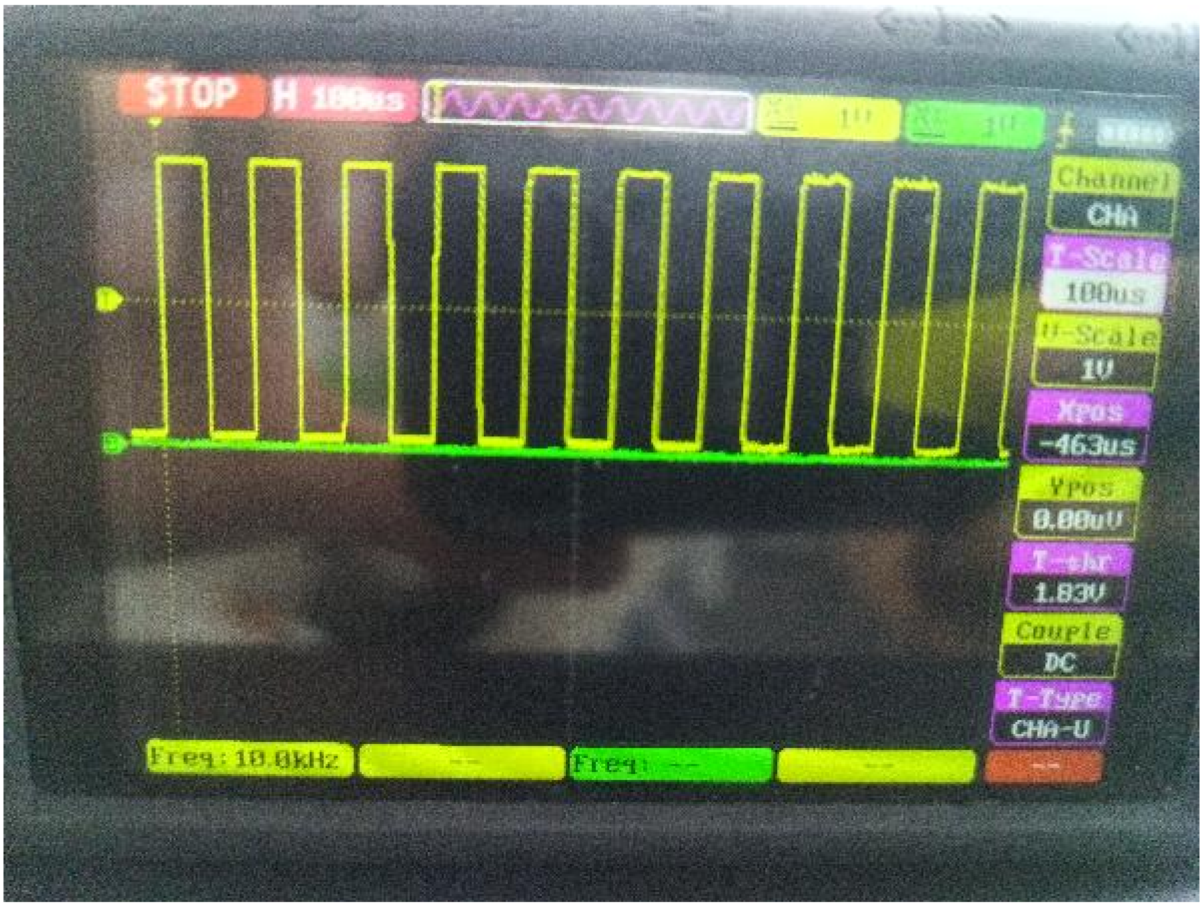
6.1.7.PWM测试

1.直接运行以下命令

```
# echo  0 > /sys/class/pwm/pwmchip0/export
# echo  0 > /sys/class/pwm/pwmchip1/export
# echo  100000 > /sys/class/pwm/pwmchip0/pwm0/period
# echo  100000 > /sys/class/pwm/pwmchip1/pwm0/period
# echo  50000 > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
# echo  50000 > /sys/class/pwm/pwmchip1/pwm0/duty_cycle
# echo  1 > /sys/class/pwm/pwmchip0/pwm0/enable
# echo  1 > /sys/class/pwm/pwmchip1/pwm0/enable
```

2.将PWM0（J8插座上 32脚）、PWM1（J8插座上 33脚）连在示波器上，查看到频率10KHz的波形


LOONGSON
G2K0321_V1.1
GPIO
HanRun



6.1.8.RTC测试

该用例将 RTC 时间+1天后，进行掉电测试，注意连接RTC电

池。 1.进入 rtc_test 目录，运行用例

```
# cd /root/loongson_test_case/rtc_test #
./rtc_test
rtc read test
RTC data/time: 2022/9/8 03:41:36
OS data/time(UTC): 2022/9/8 03:41:34
--------------
OS data/time(local): 2022/9/8 11:41:34

set rtc time test
rtc test time add one day set
rtc time
read cur rtc timu
RTC data/time: 2022/9/9
03:41:36 recover rtc time
rtc time test RW passed! #
```

2.断电几天后，重新上电，启动至系统，观察rtc时间是否多一天

6.1.9. LCD测试

1.确保开机前接上 LCD

2.进入 *lcd_test* 目录，运行用例

```
# cd /root/loongson_test_case/lcd_test #

./LCD_test
```

3.观测LCD屏幕的显示情况： LCD屏幕全屏显示红色， 一秒之后全屏显示绿色，再一秒之后显示蓝色。过 了一秒后，重复显示前三秒的效果

6.2.开发板稳定性测试

6.2.1 重启测试

可以用 systemd service 创建一个自动重启服

务。 1.创建 autoreboot.service 的文件，

写入以下内容

```
 [Unit]
Description=auto reboot test

#Documentation=man:mandb(8)

After=getty@.service



 [Service]
Type=oneshot

#ExecStart=sleep 10

ExecStart=/opt/autoreboot.sh &

 [Install]
WantedBy=multi-user.target
```

2./opt 目录下创建 autoreboot.sh，写入以下内容

```
#!/bin/bash
set -x

test_cnt=1000
sleep_time=60

date=`date +%Y%m%d`
log_dir=/opt/log_autoreboot
log_file=$log_dir/log_$date.txt
cnt_file=$log_dir/cnt.txt


if [ ! -d $log_dir ] ; then mkdir
        -p $log_dir
fi

if [ -e $cnt_file ] ; then
        test_cnt=`cat $cnt_file` fi
```

```
if [ $test_cnt -le 0 ] ; then echo
    "end .."
else
    echo "sleep $sleep_time ..." sleep
    $sleep_time
    date_str=`date +%Y%m%d%H%M%S`
    echo "$date_str  reboot test  $test_cnt ..." >> $log_file
    reboot
```

3.执行以下命令启动服务

```
# sudo chmod +x /opt/autoreboot.sh
# sudo cp autoreboot.service /lib/systemd/system #
sudo systemctl enable autoreboot
#
```

6.2.2 LTP测试

在资料包中可以得到ltp-testsuite.tar.gz的

压缩包 1.将ltp包解压在SSD或U盘介质内

2.挂载SSD或U盘（比如/media/usb 目录），并进入 ltp 路径，运行 runltp

```
# mount /dev/sda1 /media/usb
# cd /media/usb/ltp-testsuite #
./runltp
...
#
```

注:

> 1.如果想指定输出文件，可参考以下命令
>
> ```
> # ./runltp -p -l /root/log -d /tmp -o /root/printf -t 10h
> ```
>
> -l 指定结果汇总文件路
> 径 -o 指定打印信息文
> 件路径 -t 指定测试时间
> -d 指定测试的时候过程文件存放的位
> 置 -p 调整结果汇总的输出格式，便
> 于阅读

> 2.如果因为设备容量小，而将ltp放在U盘(sda1)上运行时，需要 -b 指定另外一个未挂载的块设备 (sda2)，否则部分测试项会报错
>
> ```
> # ./runltp -b /dev/sda2
> ```

6.2.3 stress测试

```
# stress --cpu 1 --io 4 --vm 2 --vm-bytes 128M --timeout 3600

stress: info: [44446] dispatching hogs: 1 cpu, 4 io, 2 vm, 0 hdd stress:

info: [44446] successful run completed in 3600s

#
```

6.2.4 memtest测试

```
# memtester 128M 1
memtester version 4.5.0 (64-bit)
Copyright (C) 2001-2020 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).


pagesize is 16384

pagesizemask is 0xffffffffffffc000

want 128MB                 bytes)
(134217728
got  128MB                          bytes), trùing mlock ...locked.
(134217728
Loop 1/1:
  Stuck Address           : ok
  Random Value            : ok
  Compar  XOR             : ok
  e
  Compar  SUB             : ok
  e
  Compar  MUL             : ok
  e
  Comtare DIV             : ok
  Coopare  OR             : ok
  Compar  AND             : ok
  e

  Sequential   Increment: ok :

  Solid Bit               ok


Done.

#
```

七、开发环境搭建

7.1.开发环境简述

广东龙芯嵌入式板卡的推荐开发环境是采用x86的ubuntu18.04交叉编译的方式进行开发。

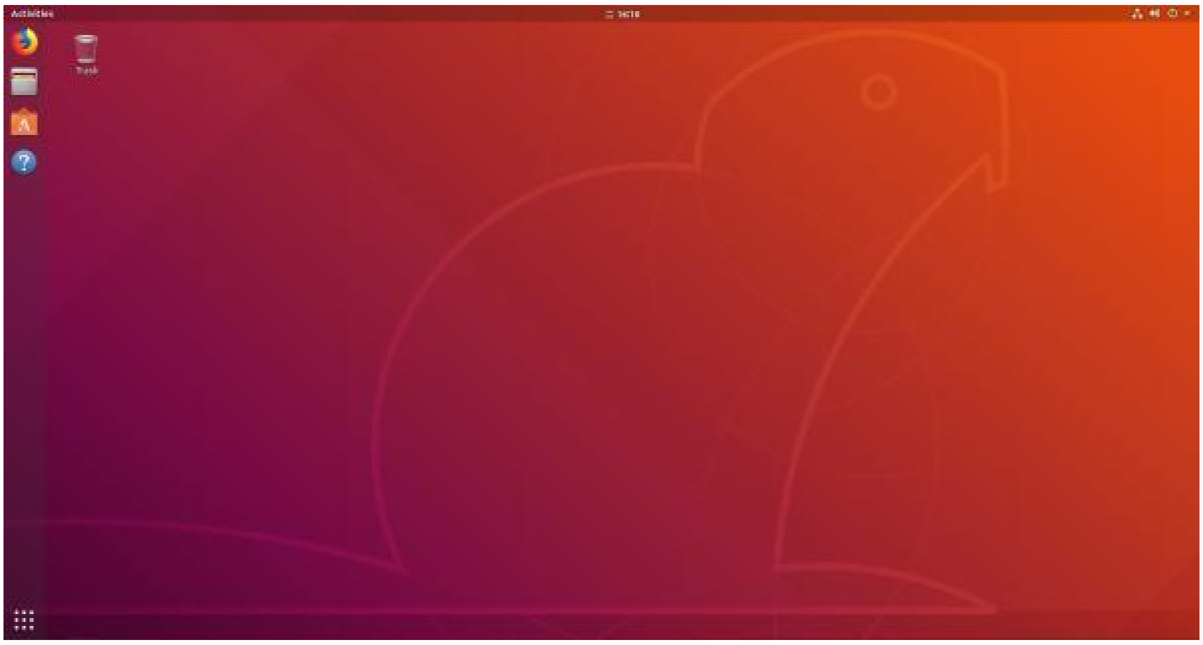
而广东龙芯的嵌入式板卡的软件组成包括固件(uboot) + 内核(linux) +文件系统。也就是说将会在x86的 ubuntu 18.04里面完成uboot、linux内核、文件系统和软件的**编译**，然后通过usb或者tftp方式**烧录**到板 卡，然后**执行对应软件**，从而完成板卡的**基本开发**。

7.2. Ubuntu18.04的配置

本节不再赘述ubuntu 18.04的安装过程，因为可以选择使用虚拟机或者直接安装，所以请自行安装完 毕。

同时在资料包里面也放置了一个配置好的ubuntu 18.04的虚拟机。可运行在VMware 15.5软件上，有条 件的话可以**直接运行此虚拟机进行开发**便**无需再次搭建环境**。**账户是loongson，密码是123** ，root账户 没设置密码。提供的虚拟机已经验证过可以完成uboot和内核的编译。

安装后ubuntu的界面如下：



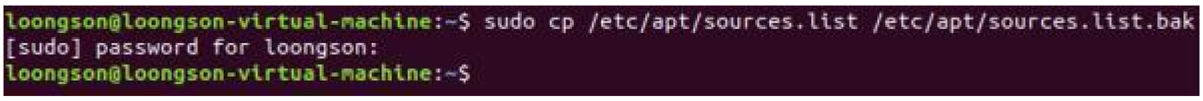
7.2.1.更新源

使用ubuntu自带的源，下载速度很慢，如果采用国内的源，那么就可以加快下载速度。具体操作如

下：可以选择其他国内源，而下面演示的是阿里源和163源。

修改/etc/apt/source.list文件，注意先保存旧的source.list文件比较稳健。

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
```



```
loongson@loongson-virtual-machine:~$ sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
[sudo] password for loongson:
loongson@loongson-virtual-machine:~$
```

```
# 添加阿里源
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe
multiverse

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to #
newer versions of the distribution.
deb http://mirrors.163.com/ubuntu bionic main restricted
# deb-src http://mirrors.163.com/ubuntu bionic main restricted

## Major bug fix updates produced after the final release of the ##
distribution.
deb http://mirrors.163.com/ubuntu bionic-updates main restricted
# deb-src http://mirrors.163.com/ubuntu bionic-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu ##
team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://mirrors.163.com/ubuntu bionic universe
# deb-src http://mirrors.163.com/ubuntu bionic universe
deb http://mirrors.163.com/ubuntu bionic-updates universe
# deb-src http://mirrors.163.com/ubuntu bionic-updates universe

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu ##
team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in ##
multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://mirrors.163.com/ubuntu bionic multiverse
# deb-src http://mirrors.163.com/ubuntu bionic multiverse
deb http://mirrors.163.com/ubuntu bionic-updates multiverse
# deb-src http://mirrors.163.com/ubuntu bionic-updates multiverse
```

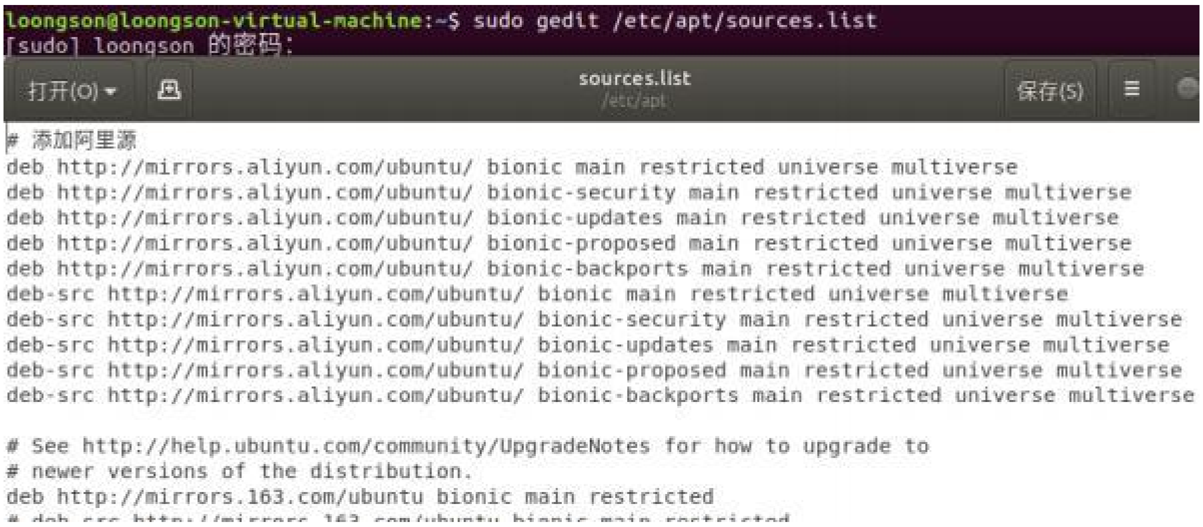
```
## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes ## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review ## or updates from the Ubuntu security team.
deb http://mirrors.163.com/ubuntu bionic-backports main restricted universe multiverse
# deb-src http://mirrors.163.com/ubuntu bionic-backports main restricted universe multiverse
```

```
deb http://mirrors.163.com/ubuntu bionic-security main restricted
# deb-src http://mirrors.163.com/ubuntu bionic-security main restricted
deb http://mirrors.163.com/ubuntu bionic-security universe
# deb-src http://mirrors.163.com/ubuntu bionic-security universe
deb http://mirrors.163.com/ubuntu bionic-security multiverse
# deb-src http://mirrors.163.com/ubuntu bionic-security multiverse
```

输入命令：

```
sudo gedit /etc/apt/sources.list
```

然后编辑器里面把原来的内容全部替换为上文所述的内容。



```
loongson@loongson-virtual-machine:~$ sudo gedit /etc/apt/sources.list
[sudo] loongson 的密码:
```

```
# 添加阿里源
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://mirrors.163.com/ubuntu bionic main restricted
# deb-src http://mirrors.163.com/ubuntu bionic main restricted
```

随后输入命令：（保证网络畅通）

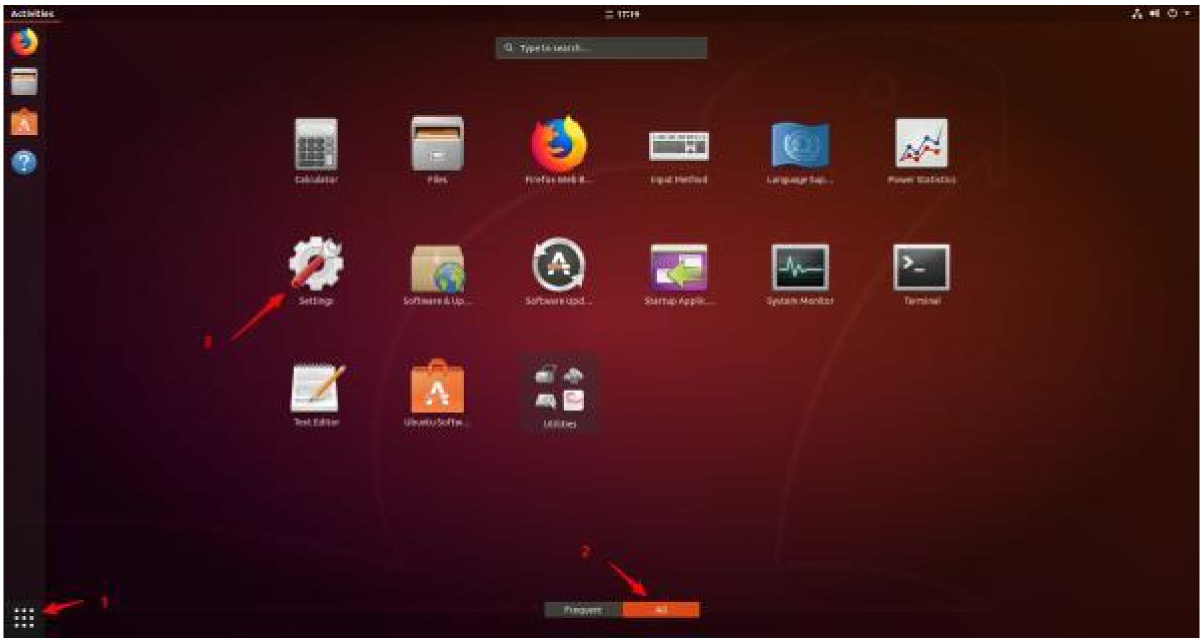
```
sudo apt update
```

等待后即可完成源的更新。

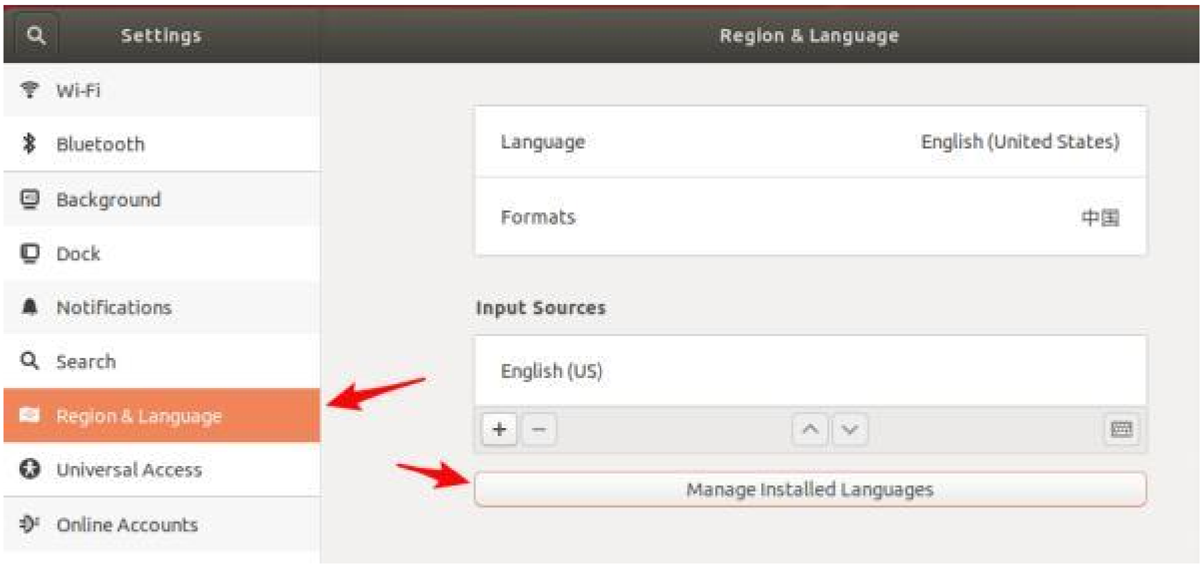
7.2.2.安装中文环境

如果安装时选择英文安装，那么安装完毕之后，是英文界面的。如果想修改为中文界面，可以参考以下步骤：

打开设置：



选择语言设置：

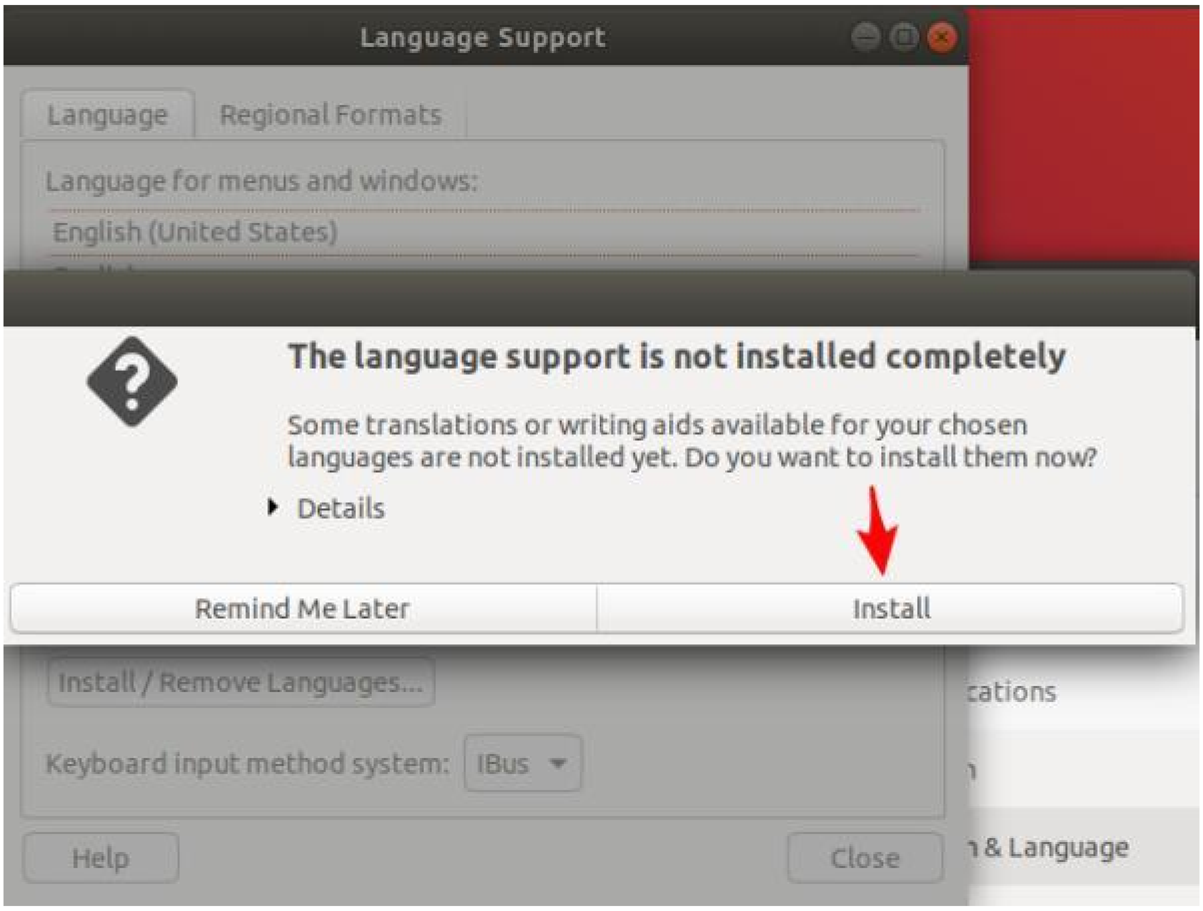


Language English (United States)

Formats 中国

English (US)

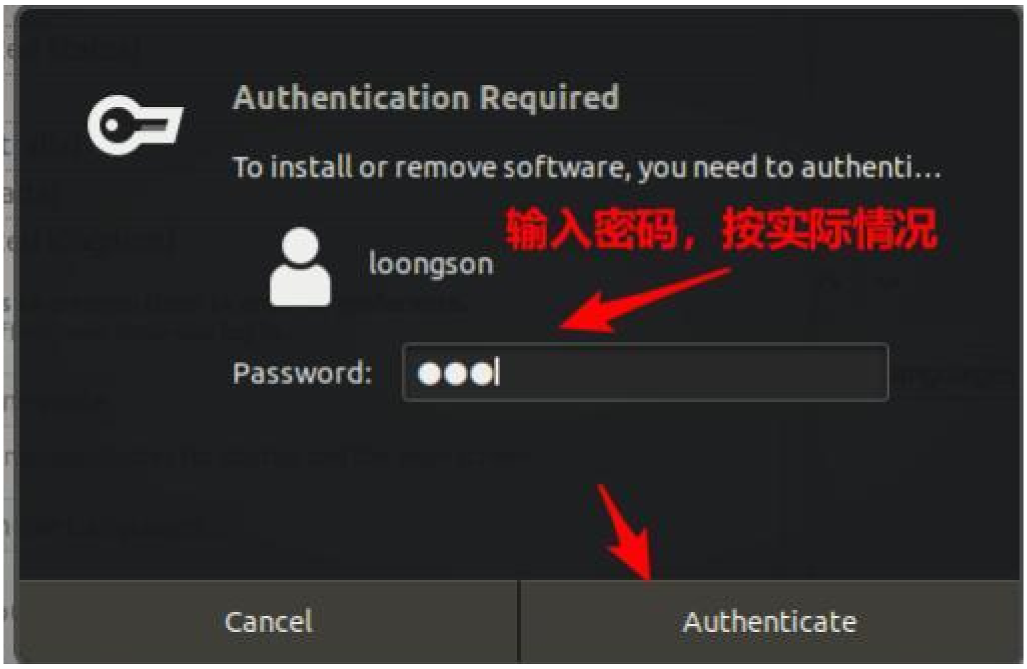
下载字体：



The language support is not installed completely

Some translations or writing aids available for your chosen languages are not installed yet. Do you want to install them now?

输入当前用户密码：

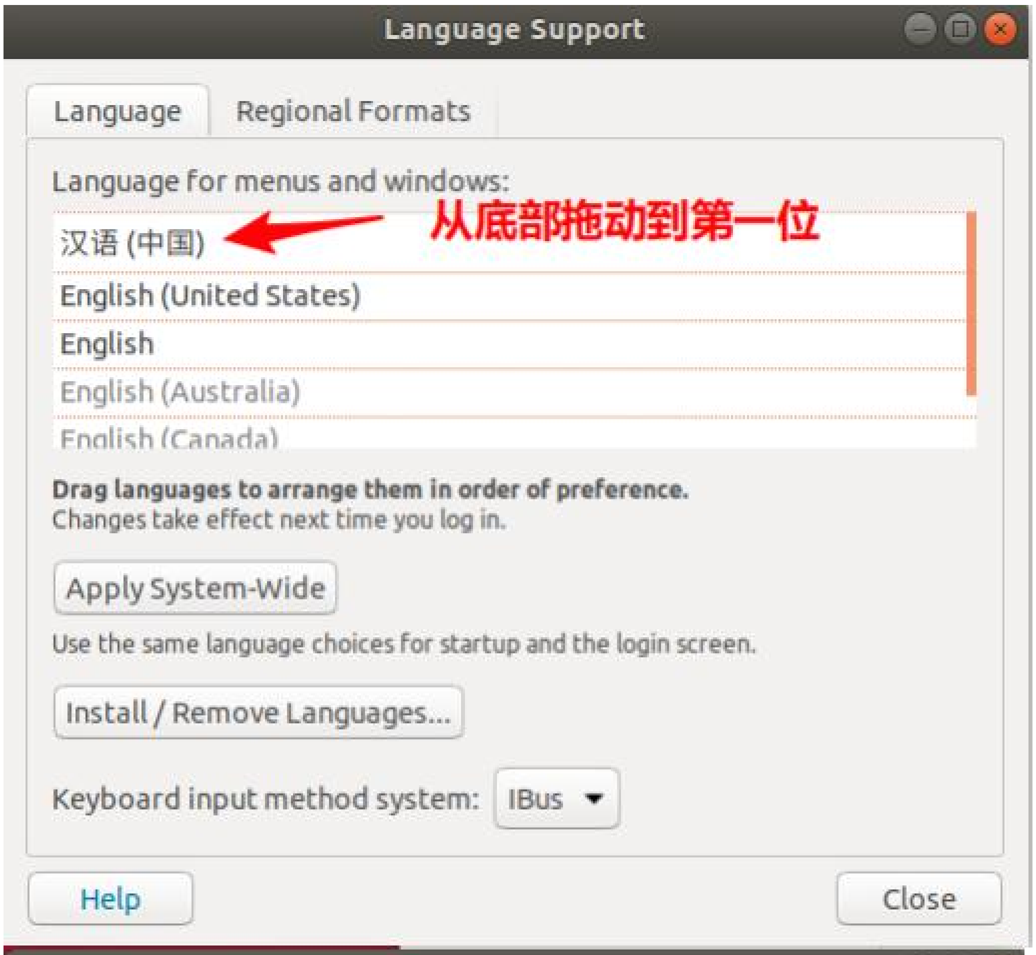


Authentication Required

To install or remove software, you need to authenti…

等待下载完毕：

随后按照下图操作



Language Support

Language | Regional Formats

Language for menus and windows:

汉语 (中国)
English (United States)
English
English (Australia)
English (Canada)

从底部拖动到第一位

Drag languages to arrange them in order of preference.
Changes take effect next time you log in.

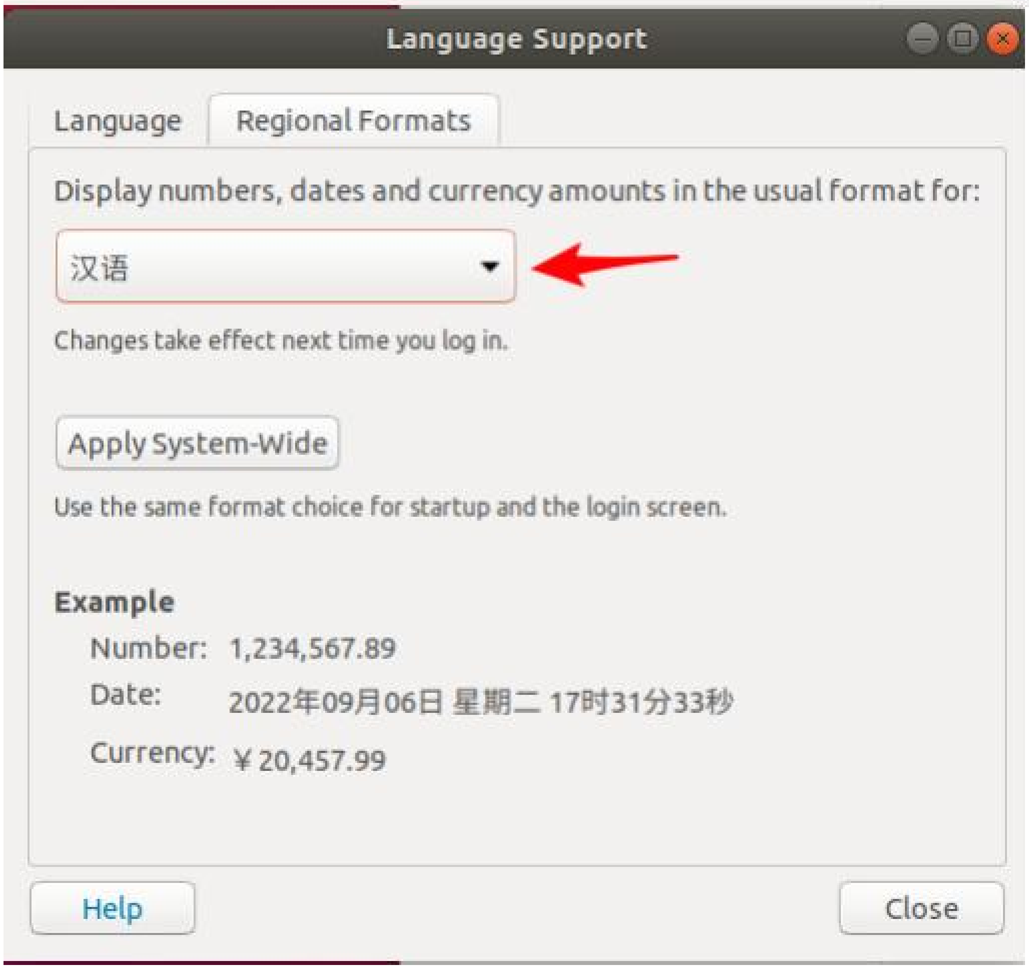
Apply System-Wide

Use the same language choices for startup and the login screen.

Install / Remove Languages...

Keyboard input method system: IBus

Help Close



Language Support

Language | Regional Formats

Display numbers, dates and currency amounts in the usual format for:

汉语

Changes take effect next time you log in.

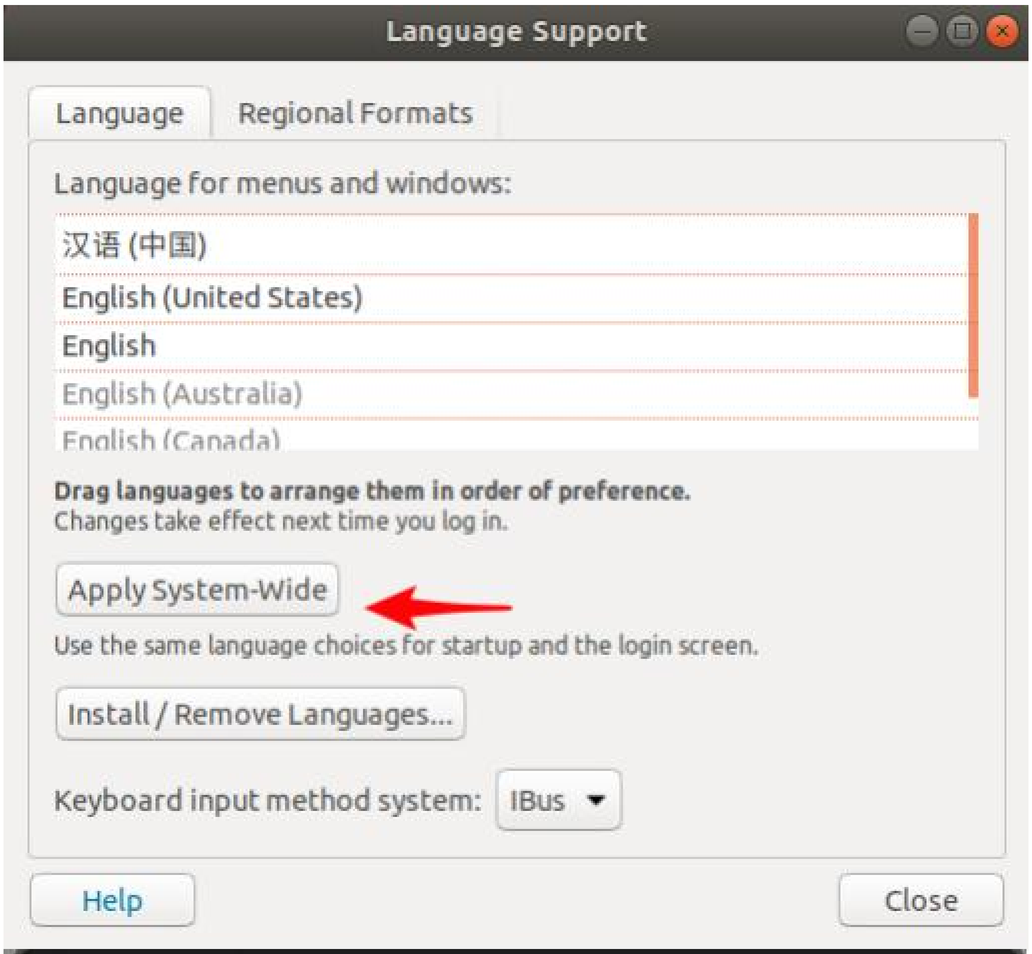
Apply System-Wide

Use the same format choice for startup and the login screen.

Example

Number: 1,234,567.89
Date: 2022年09月06日 星期二 17时31分33秒
Currency: ￥20,457.99

Help Close


Language Support

Language | Regional Formats

Language for menus and windows:

汉语 (中国)
English (United States)
English
English (Australia)
English (Canada)

Drag languages to arrange them in order of preference.
Changes take effect next time you log in.

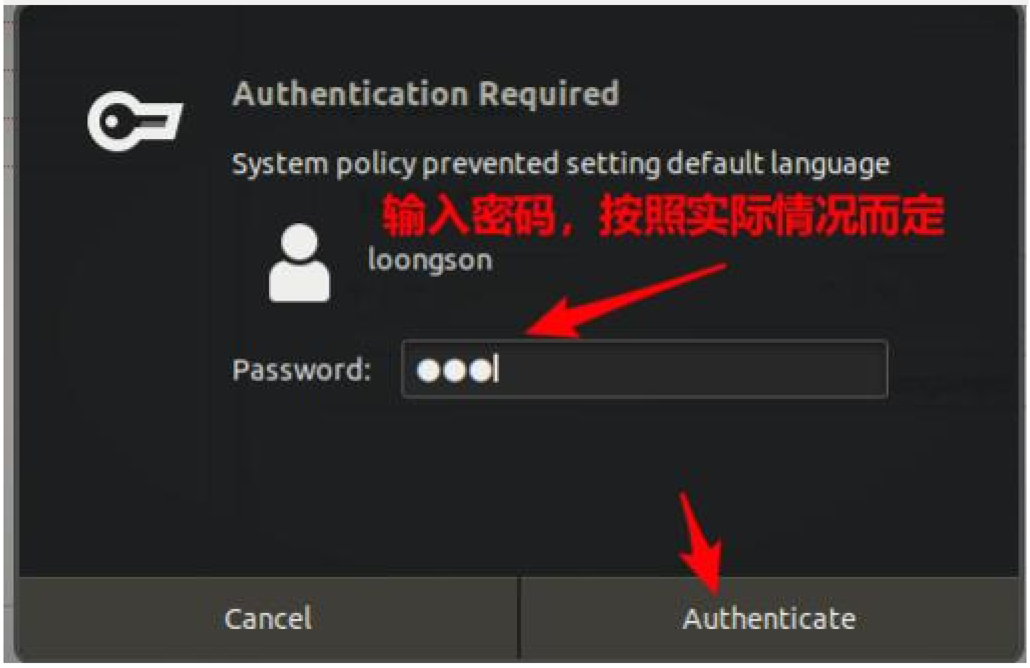
Apply System-Wide

Use the same language choices for startup and the login screen.

Install / Remove Languages...

Keyboard input method system: IBus

Help | Close


Authentication Required

System policy prevented setting default language

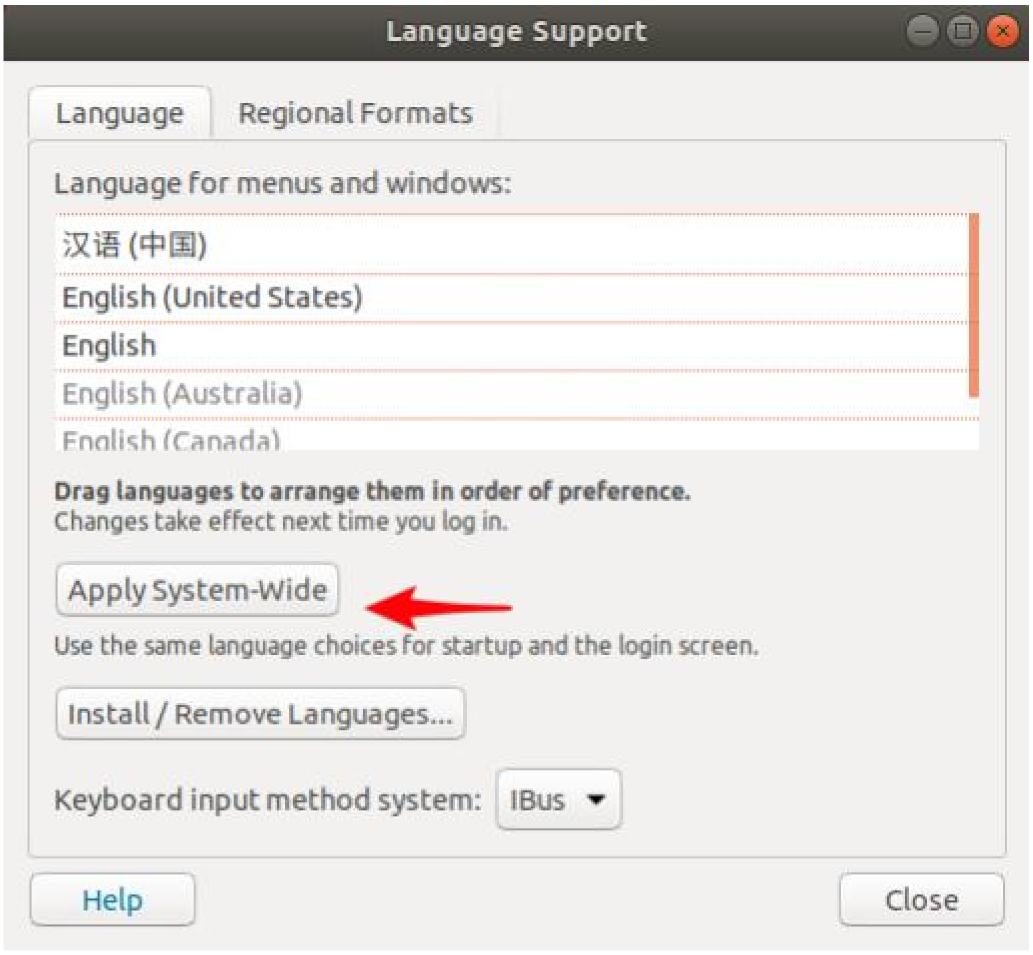
输入密码，按照实际情况而定

loongson

Password: ●●●

Cancel | Authenticate

应用后重启机器，重启后出现下面的弹窗提示，推荐按照下面的提示执行：


Language Support
Language
Regional Formats
Language for menus and windows:
汉语 (中国)
English (United States)
English
English (Australia)
English (Canada)
Drag languages to arrange them in order of preference.
Changes take effect next time you log in.
Apply System-Wide
Use the same language choices for startup and the login screen.
Install / Remove Languages...
Keyboard input method system:
IBus
Help
Close

7.2.3.部署开发环境

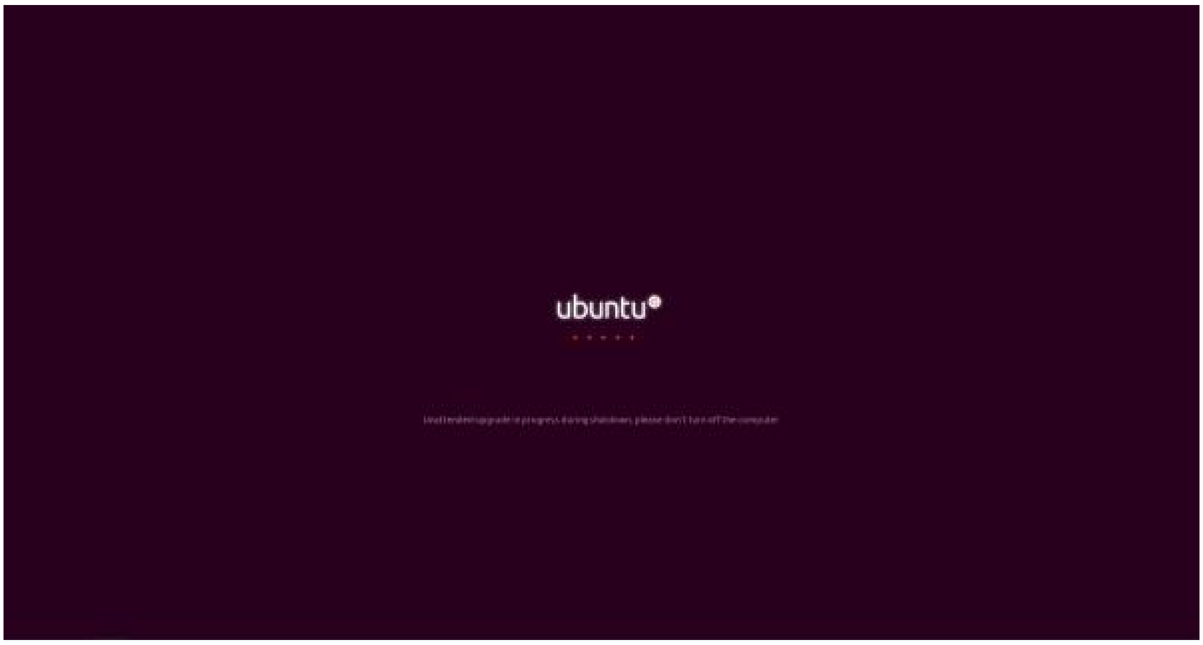
本节将会使用apt安装固件、内核、文件系统和相关开发用的软件

。 执行命令：

```
sudo apt -y install make git gcc g++ bison flex libncurses5-dev libssl-dev libelf-dev u-
boot-tools
sudo apt -y install cmake tree build-essential tcl-dev automake libtool
```

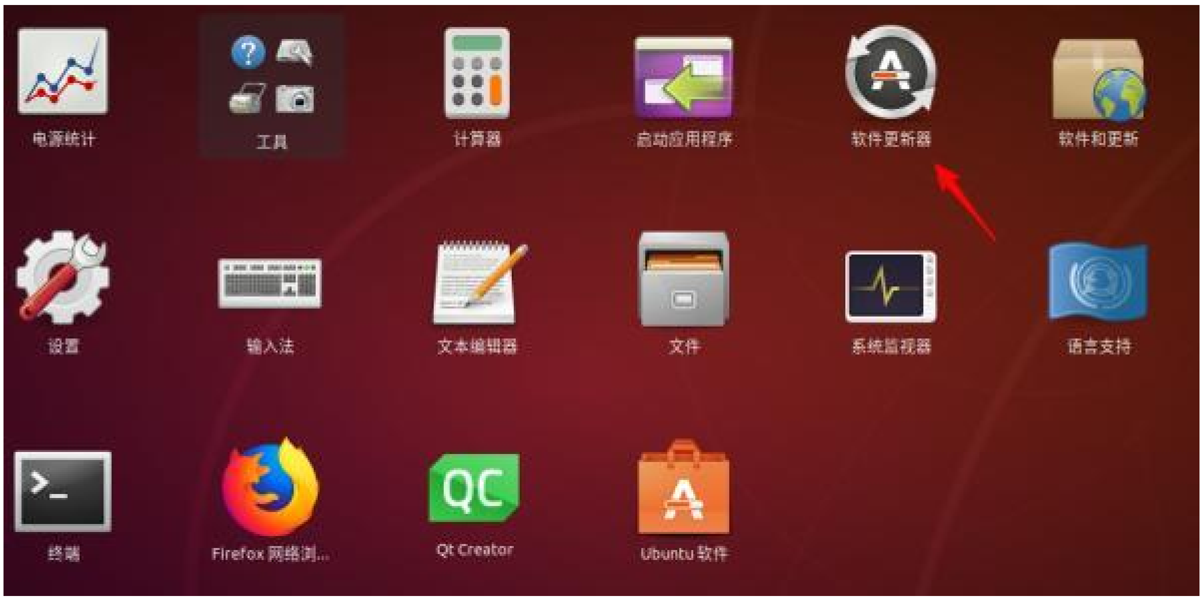
7.2.4.关闭自动更新

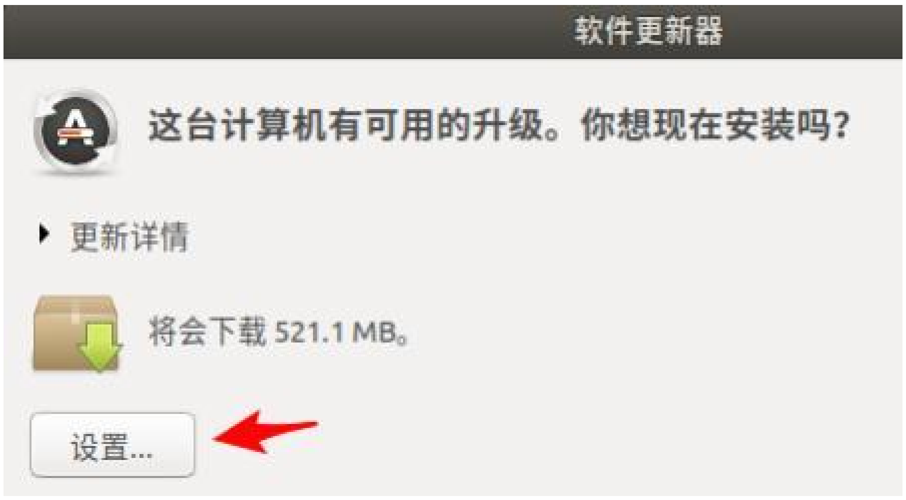
如果发现ubuntu 18.04在关机时卡在下面展示的页面很久。



Unattended-upgrade in progress during shutdown, please don't turn off the computer

可以参考下面的操作解决

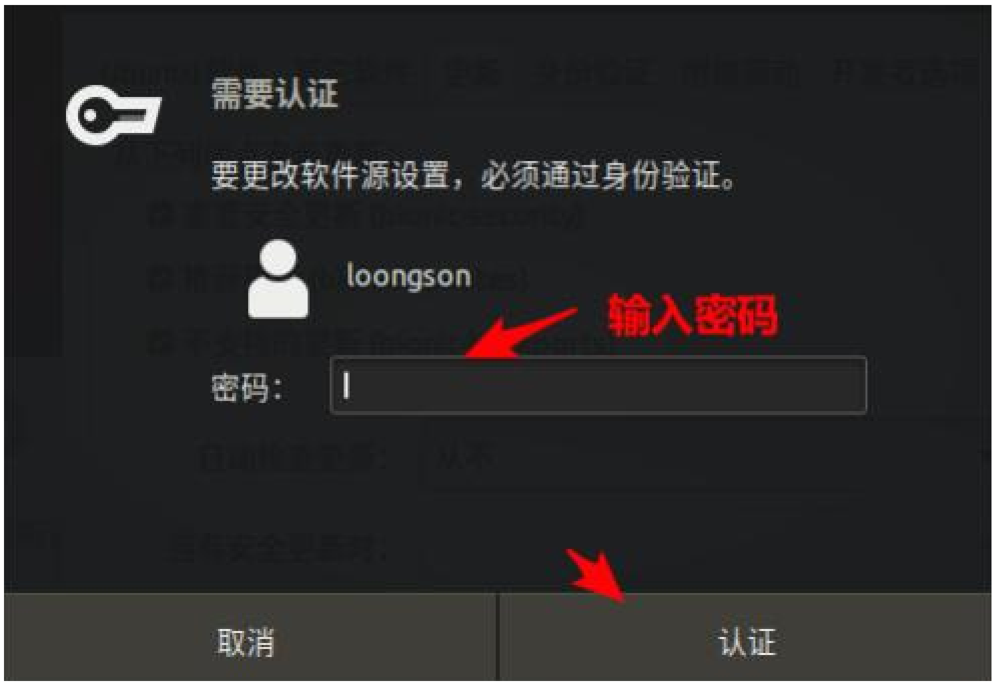



软件更新器

这台计算机有可用的升级。你想现在安装吗?

更新详情

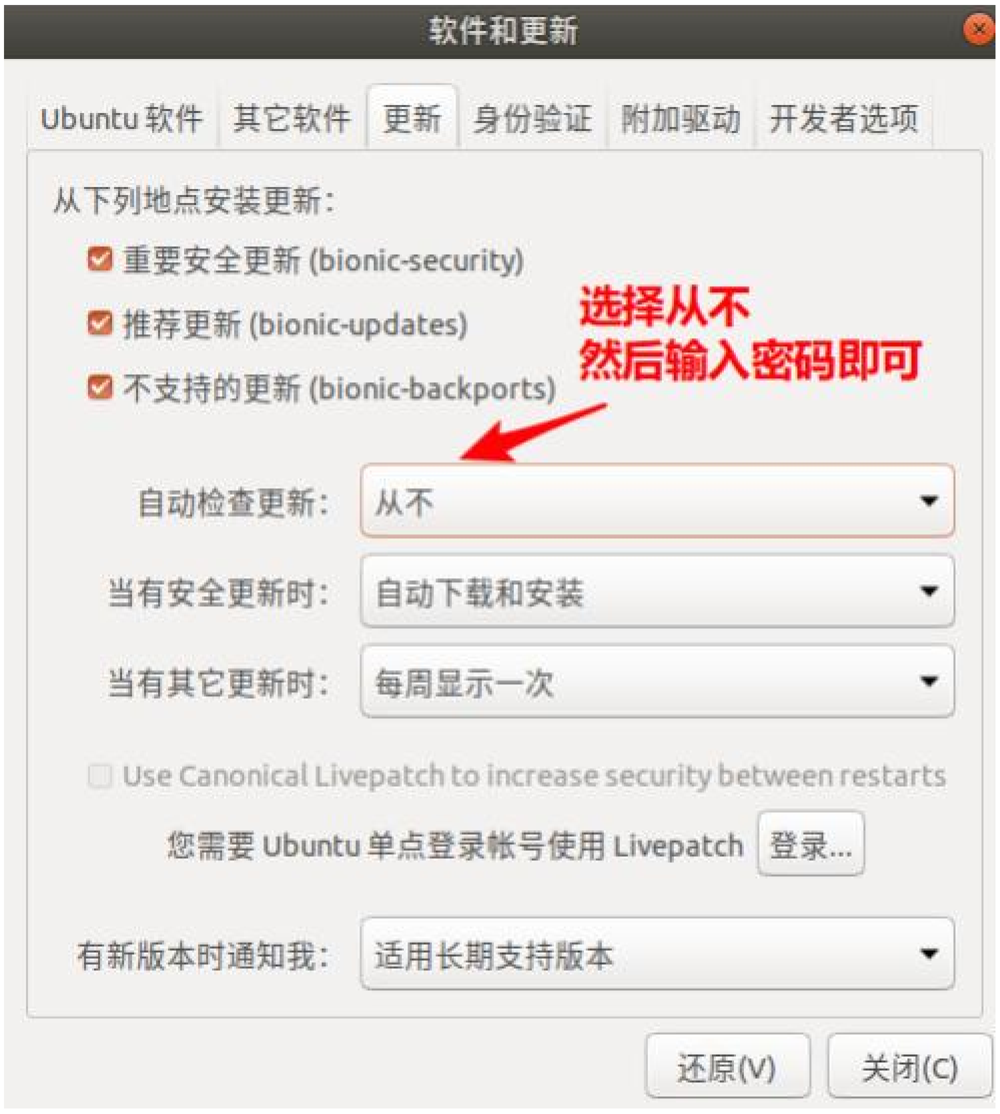
将会下载 521.1 MB。


需要认证

要更改软件源设置，必须通过身份验证。

loongson

密码：



从下列地点安装更新：

- ☑ 重要安全更新 (bionic-security)
- ☑ 推荐更新 (bionic-updates)
- ☑ 不支持的更新 (bionic-backports)

自动检查更新： 从不

当有安全更新时： 自动下载和安装

当有其它更新时： 每周显示一次

☐ Use Canonical Livepatch to increase security between restarts

您需要 Ubuntu 单点登录帐号使用 Livepatch

有新版本时通知我： 适用长期支持版本

本质上是每次关机的时候，都会检查软件更新，只需要关闭这项功能即可。

7.3.交叉编译工具链安装

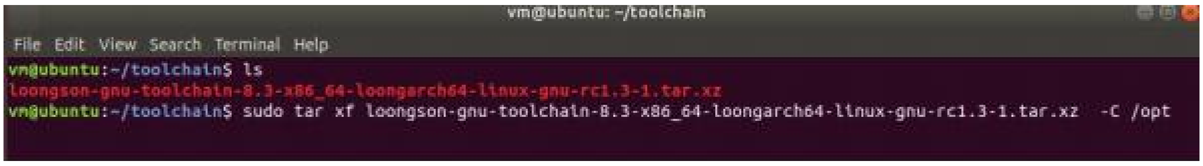
目前loongarch64所用的交叉编译工具链的包为： `loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu-rc1.3-1.tar.xz`

这个压缩包能够在资料包中找到。

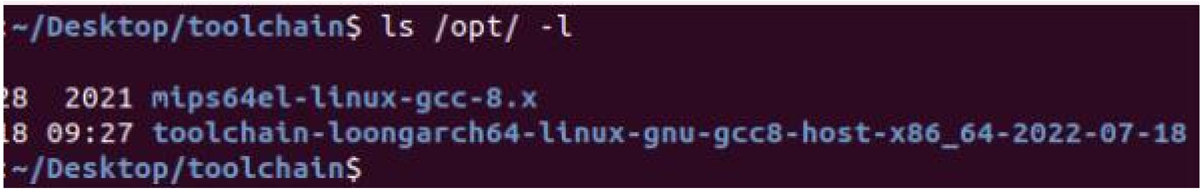
把压缩包传输到ubuntu18.04里面。下面的演示放在了桌面的toolchain文件夹中，这只是演示，可根据 实际情况调整。

安装工具链的过程其实就是把压缩包解压到/opt目录下。

```
sudo tar -xf loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu-rc1.3-1.tar.xz -C /opt/
```



```
vm@ubuntu:~/toolchain$ ls
loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu-rc1.3-1.tar.xz
vm@ubuntu:~/toolchain$ sudo tar xf loongson-gnu-toolchain-8.3-x86_64-loongarch64-linux-gnu-rc1.3-1.tar.xz  -C /opt
```



```
:~/Desktop/toolchain$ ls /opt/ -l

28  2021 mips64el-linux-gcc-8.x
18 09:27 toolchain-loongarch64-linux-gnu-gcc8-host-x86_64-2022-07-18
:~/Desktop/toolchain$
```

7.4.关于buildroot的编译

对于buildroot的编译，完整编译一个文件系统后， buildroot的体积将会达到12G以上。如果是使用虚拟 机的情况下，那么虚拟机的体积也会对应膨胀。如果个人电脑的性能不足，完整编译时间将会达到5小时 以上。如果有服务器的话，将会把时间减少到半小时以内（24核服务器为例）。

buildroot的编译请查看 [9.1.1 buildroot编译]一节

7.5.关于loongarch交叉编译autotool出错的问题

很多软件是autotools-package的形式构建，通俗点说就是要执行./configure的那些软件。如果是要进行 loongarch的交叉编译。由于loongarch是新的架构，大多数旧的系统还没支持。所以/usr/share/misc的 config.guess和 config.sub里面没有loongarch的记录， 从而导致configure执行失败。

解决办法请参考[8.5 使用autotools-package编译（./configure）]一节， 里面有描述如何修改ubuntu 18.04的 `config.guess`和 `confg.sub`。

八、应用开发

8.1.交叉编译软件

对于busybox系统。由于没有编译系统在里面，所以如果需要软件在板卡上运行，那么交叉编译是必不 可少的。

如果对交叉编译(cross compile)这个概念不太清楚的话，可以参考以下的一个说法。交叉编译通常指的 是在PC机上使用交叉编译工具链（通常是编译内核的那个工具链），完成软件的编译，编译出一个可执 行程序，此程序是基于对应CPU的架构上才能运行。这种情况叫做交叉编译。

对于龙芯嵌入式系列板卡，则采用以下方式进行交叉编译：在**X86机器的ubuntu18.04**的系统中，使用 交叉编译工具链，编译软件，然后复制编译出来的可执行文件到板卡中（USB传输等手段），然后才能 在板卡上运行。

需要注意的是，交叉编译出来的软件，不仅仅可以给busybox用， loongnix系统也能用。因为编译出来 的可执行程序是适应对应架构的。

下文将会介绍软件的交叉编译方式，其中包括一般软件和Qt软件。

一般软件，本处的定义通常指代为没有使用特殊IDE进行编译的软件。下面将会介绍， **手动编译，使用 makefile编译，使用Cmake编译，对于autotools-package方式的源码编译（有configure文件那 种）**。

由于此章节适用于多数嵌入式板卡，但是编译和cpu的架构相关，所以会有所差别，下文中的编译演示将 会 以 loongarch来 说 明

不同的架构，只是采用的交叉编译工具链不同，然后导致声明交叉编译工具链的命令不同。（下面的演示不包含工具链的部署）

下面的演示中，声明交叉编译工具链的命令是3条export语句。

例如：对于loongarch64（即64位loongarch架构)目前用的交叉编译工具链是loongson-gnu-toolchain- 8.3-x86_64-loongarch64-linux-gnu-rc1.3-1.tar.gz

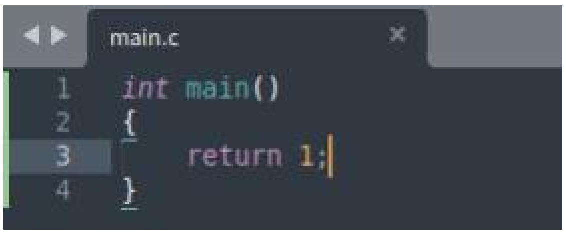
命令如下：

```
export PATH=$PATH:/opt/loongson-gnu-toolchain-8.3-x86_64-

loongarch64-linux-gnu-rc1.3-1/bin/ export ARCH=loongarch64

export CROSS_COMPILE=loongarch64-linux-gnu-
```

工具链可根据实际情况而定，声明工具链的PATH那个命令，路径是去到工具链里面的bin目录即可。

8.2.手动编译软件

以一段简单的代码用以说明，代码如下：

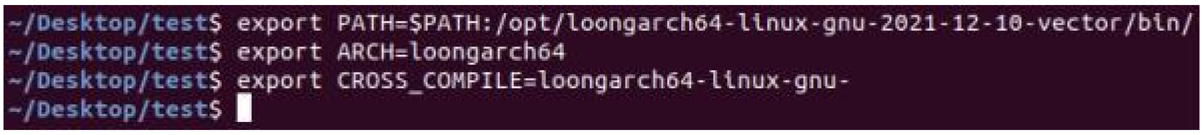

```
int main()
{
    return 1;
}
```

然后声明工具链，即在shell终端输入以下命令：（这个只是例子，按实际情况而定）

export PATH=$PATH:/opt/loongson-gnu-toolchain-8.3-x86_64-

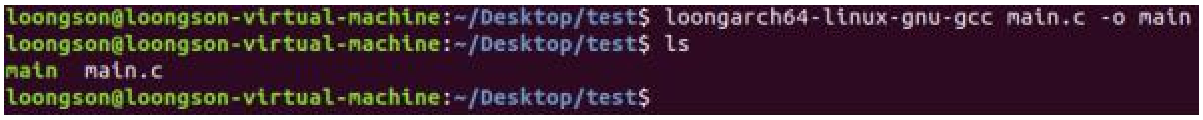
loongarch64-linux-gnu-rc1.3-1/bin/ export ARCH=loongarch64

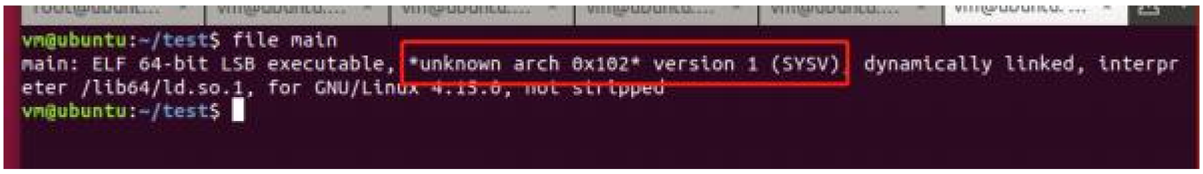
export CROSS_COMPILE=loongarch64-linux-gnu-


```
~/Desktop/test$ export PATH=$PATH:/opt/loongarch64-linux-gnu-2021-12-10-vector/bin/
~/Desktop/test$ export ARCH=loongarch64
~/Desktop/test$ export CROSS_COMPILE=loongarch64-linux-gnu-
~/Desktop/test$
```

随后输入以下命令进行编译：（这个只是例子，按实际情况而定）。

loongarch64-linux-gnu-gcc main.c -o main


```
loongson@loongson-virtual-machine:~/Desktop/test$ loongarch64-linux-gnu-gcc main.c -o main
loongson@loongson-virtual-machine:~/Desktop/test$ ls
main  main.c
loongson@loongson-virtual-machine:~/Desktop/test$
```


```
vm@ubuntu:~/test$ file main
main: ELF 64-bit LSB executable, *unknown arch 0x102* version 1 (SYSV), dynamically linked, interpr
eter /lib64/ld.so.1, for GNU/Linux 4.15.0, not stripped
vm@ubuntu:~/test$
```

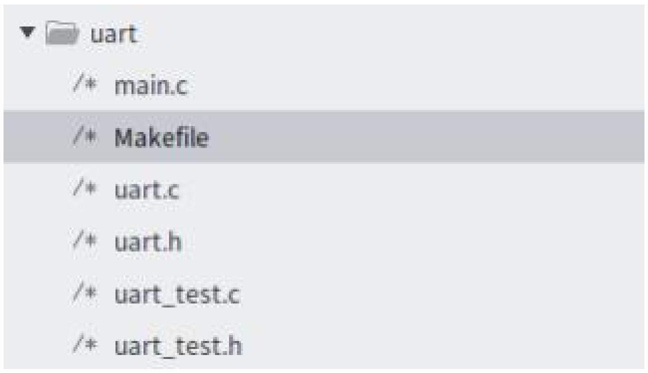
通过file命令查看编译后的main 文件，显示 *unknown arch 0x102*，这是因为LoongArch的二进制编 号为258(0x102)，较新的file 命令才添加了对LoongArch的支持。

然后就可以复制main这个可执行程序到板卡（USB传输等方式，视情况而定），然后就能运行这个程 序。

8.3.使用Makefile编译软件

本处不再赘述Makefile的相关概念。

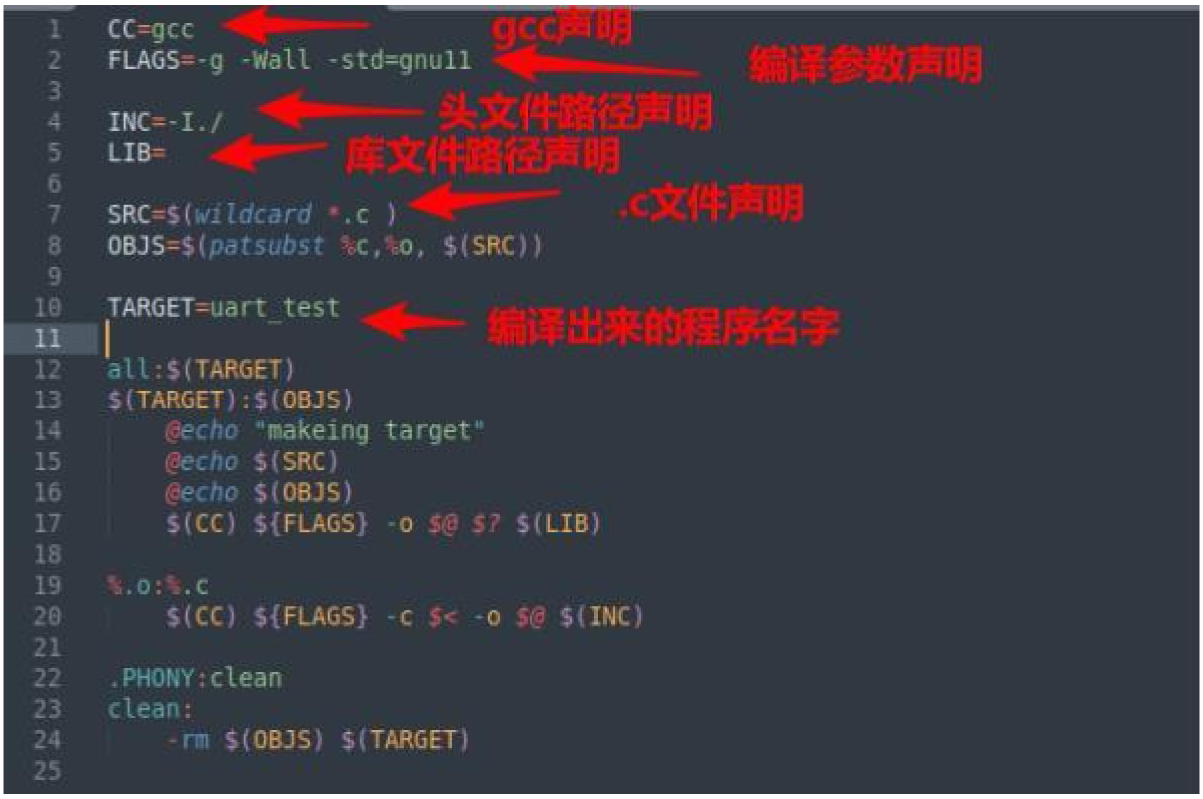
下面将会使用龙芯测试软件的一个例子来说明。



上图是软件的代码结构展示图。 main.c调用uart_test模块， uart_test调用uart模块。 下面将会提供一个Makefile文件的例子：

```
CC=gcc
FLAGS=-g -Wall -std=gnu11

INC=-I./
LIB=

SRC=$(wildcard *.c )
OBJS=$(patsubst %c,%o, $(SRC))

TARGET=uart_test

all:$(TARGET)
$(TARGET):$(OBJS)
	@echo "makeing target"
	@echo $(SRC)
	@echo $(OBJS)
	$(CC) ${FLAGS} -o $@ $? $(LIB)

%.o:%.c
	$(CC) ${FLAGS} -c $< -o $@ $(INC)

.PHONY:clean
clean:
	-rm $(OBJS) $(TARGET)
```

请注意，由于makefile对缩进的格式很严谨，上述的内容只是作为例子，如果直接使用，还需**根据实际 情况，手动调整格式**。见下图，需要使用tab进行缩进。

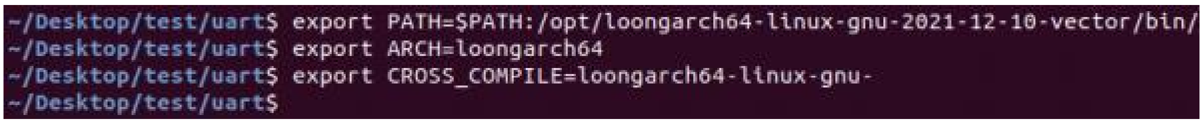


```
CC=gcc
FLAGS=-g -Wall -std=gnu11

INC=-I./
LIB=

SRC=$(wildcard *.c )
OBJS=$(patsubst %c,%o, $(SRC))

TARGET=uart_test

all:$(TARGET)
$(TARGET):$(OBJS)
	@echo "makeing target"
	@echo $(SRC)
	@echo $(OBJS)
	$(CC) ${FLAGS} -o $@ $? $(LIB)

%.o:%.c
	$(CC) ${FLAGS} -c $< -o $@ $(INC)

.PHONY:clean
clean:
	-rm $(OBJS) $(TARGET)

```

在makefile文件所在的文件夹打开终端。

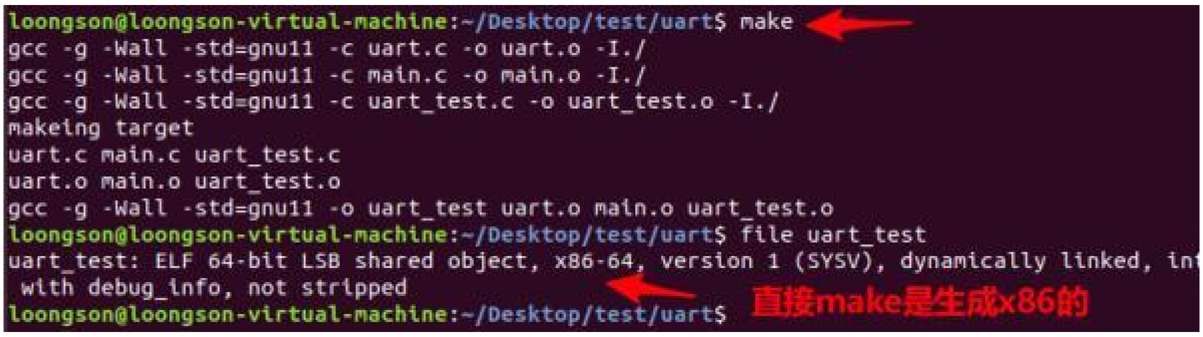
然后声明工具链，即在shell终端输入以下命令：（这个只是例子，按实际情况而定）

```
export PATH=$PATH:/opt/loongson-gnu-toolchain-8.3-x86_64-

loongarch64-linux-gnu-rc1.3-1/bin/ export ARCH=loongarch64

export CROSS_COMPILE=loongarch64-linux-gnu-
```



```
~/Desktop/test/uart$ export PATH=$PATH:/opt/loongarch64-linux-gnu-2021-12-10-vector/bin/
~/Desktop/test/uart$ export ARCH=loongarch64
~/Desktop/test/uart$ export CROSS_COMPILE=loongarch64-linux-gnu-
~/Desktop/test/uart$
```

需要注意的是，如果直接make，按照Makfile文件里面的CC=gcc，那么生成的程序就是编译的机器上能 运行的，见下图：



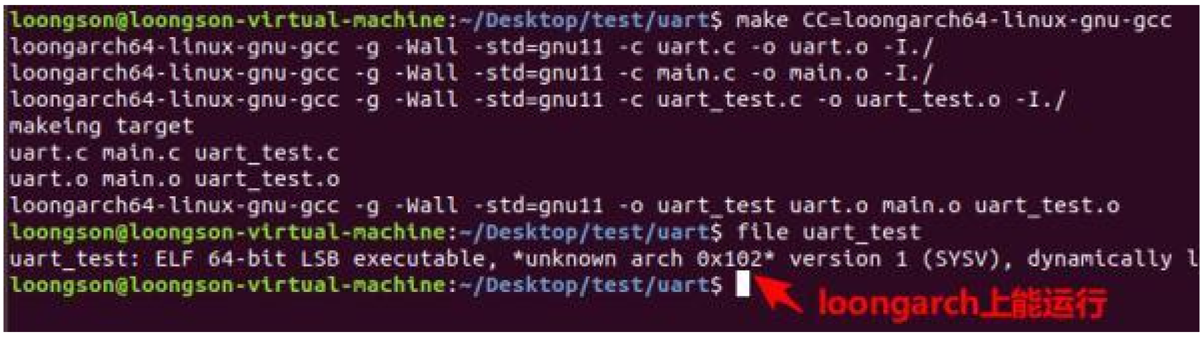
```
loongson@loongson-virtual-machine:~/Desktop/test/uart$ make
gcc -g -Wall -std=gnu11 -c uart.c -o uart.o -I./
gcc -g -Wall -std=gnu11 -c main.c -o main.o -I./
gcc -g -Wall -std=gnu11 -c uart_test.c -o uart_test.o -I./
makeing target
uart.c main.c uart_test.c
uart.o main.o uart_test.o
gcc -g -Wall -std=gnu11 -o uart_test uart.o main.o uart_test.o
loongson@loongson-virtual-machine:~/Desktop/test/uart$ file uart_test
uart_test: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, in
 with debug_info, not stripped
loongson@loongson-virtual-machine:~/Desktop/test/uart$
```

可以修改CC的值为对应交叉编译工具链gcc的名字。

源码可以在有编译条件下的系统上可以编译，比如loongnix上面有gcc，那么就能编

译。 交叉编译的时候，建议make命令为：

```
make CC=loongarch64-linux-gnu-gcc
```



```
loongson@loongson-virtual-machine:~/Desktop/test/uart$ make CC=loongarch64-linux-gnu-gcc
loongarch64-linux-gnu-gcc -g -Wall -std=gnu11 -c uart.c -o uart.o -I./
loongarch64-linux-gnu-gcc -g -Wall -std=gnu11 -c main.c -o main.o -I./
loongarch64-linux-gnu-gcc -g -Wall -std=gnu11 -c uart_test.c -o uart_test.o -I./
makeing target
uart.c main.c uart_test.c
uart.o main.o uart_test.o
loongarch64-linux-gnu-gcc -g -Wall -std=gnu11 -o uart_test uart.o main.o uart_test.o
loongson@loongson-virtual-machine:~/Desktop/test/uart$ file uart_test
uart_test: ELF 64-bit LSB executable, *unknown arch 0x102* version 1 (SYSV), dynamically l
loongson@loongson-virtual-machine:~/Desktop/test/uart$
```

同样地，复制可执行程序到板卡上运行即可。

8.4.使用Cmake编译

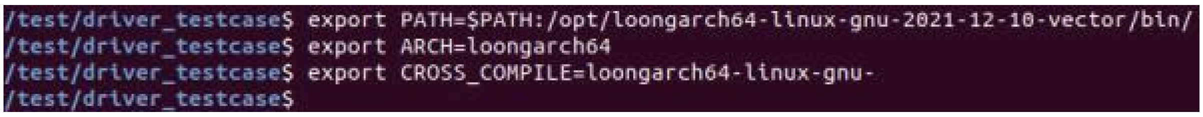
关于CMake的构建不再赘述。本处只说明cmake命令运行时如何指定交叉编译工具

链。 下面将会以龙芯测试软件的cmake构建来作为例子说明。

声明工具链，即在shell终端输入以下命令：（这个只是例子，按实际情况而定）

```
export PATH=$PATH:/opt/loongson-gnu-toolchain-8.3-x86_64-
loongarch64-linux-gnu-rc1.3-1/bin/
```

```
export ARCH=loongarch64

export CROSS_COMPILE=loongarch64-linux-gnu-
```



```
/test/driver_testcase$ export PATH=$PATH:/opt/loongarch64-linux-gnu-2021-12-10-vector/bin/
/test/driver_testcase$ export ARCH=loongarch64
/test/driver_testcase$ export CROSS_COMPILE=loongarch64-linux-gnu-
/test/driver_testcase$
```

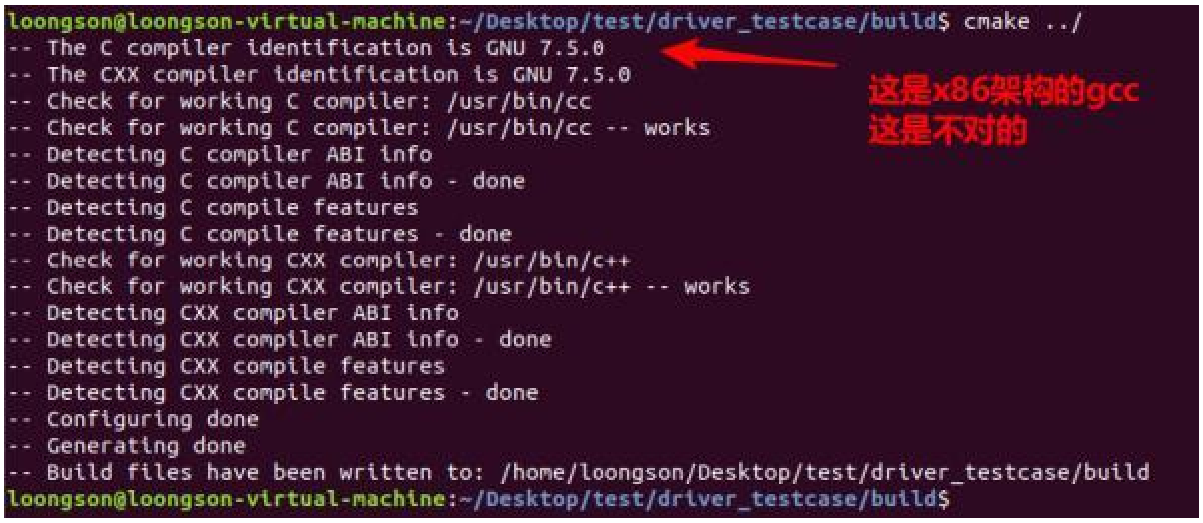
预期的构建方式是CMakeLists.txt文件在源码根目录，需要创建一个build文件夹，然后在build文件夹中 使用命令：

```
cmake ../
```

即可完成Cmake部署，然后make进行构建。

注意上述只是一个例子，构建过程需要按照实际情况调整。

如同上述的Makefile那样，没有指定对应的gcc，那么是以编译的机器的架构来编译的。

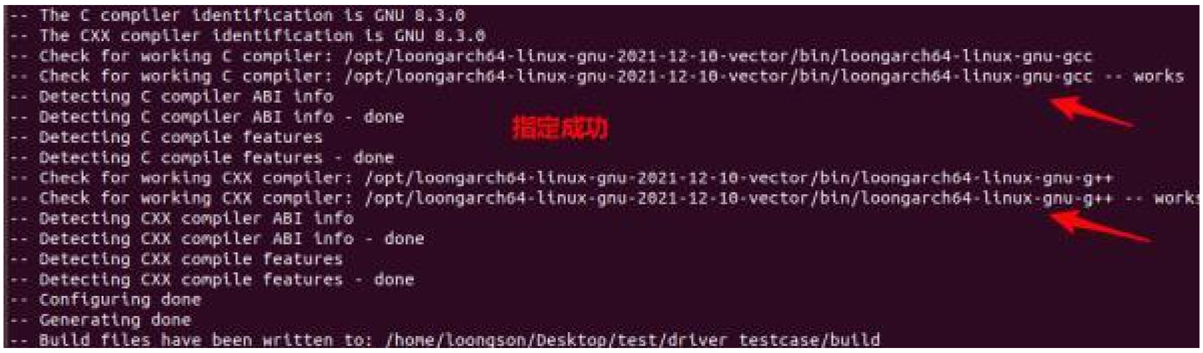


```
loongson@loongson-virtual-machine:~/Desktop/test/driver_testcase/build$ cmake ../
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/loongson/Desktop/test/driver_testcase/build
loongson@loongson-virtual-machine:~/Desktop/test/driver_testcase/build$
```

如果cmake的时候不小心没指定gcc，那么建议清空相关文件（按照本例子，则删除build文件夹），重新 操作。

那么如果要指定gcc和g++，可以参考下面的命令：

```
cmake -DCMAKE_C_
COMPILER=loongarch64-linux-gnu-gcc -
DCMAKE_CXX_COMPILER=loongarch64-linux
-gnu-g++ ../
```



```
-- The C compiler identification is GNU 8.3.0
-- The CXX compiler identification is GNU 8.3.0
-- Check for working C compiler: /opt/loongarch64-linux-gnu-2021-12-10-vector/bin/loongarch64-linux-gnu-gcc
-- Check for working C compiler: /opt/loongarch64-linux-gnu-2021-12-10-vector/bin/loongarch64-linux-gnu-gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /opt/loongarch64-linux-gnu-2021-12-10-vector/bin/loongarch64-linux-gnu-g++
-- Check for working CXX compiler: /opt/loongarch64-linux-gnu-2021-12-10-vector/bin/loongarch64-linux-gnu-g++ -- work
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/loongson/Desktop/test/driver testcase/build
```

随后make的话就是用指定的gcc编译。

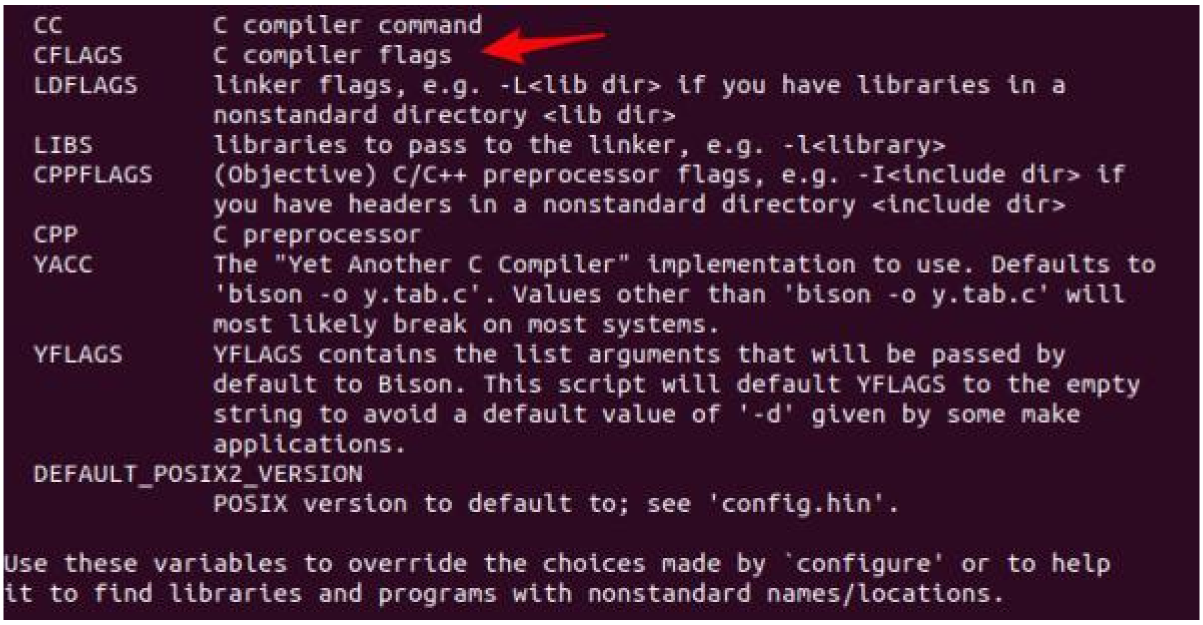
当然上述的指定gcc的方式只是其中一种，更多的方式可以根据Cmake的特性进行修改（比如修改 CMakeLists.txt文件）。

8.5.使用autotools-package编译（./configure）

对于使用configure文件来检查编译环境的源码，如果是本地编译，那么只需要./configure，make， make install即可。但是在交叉编译的时候，是需要指定交叉编译工具链的。

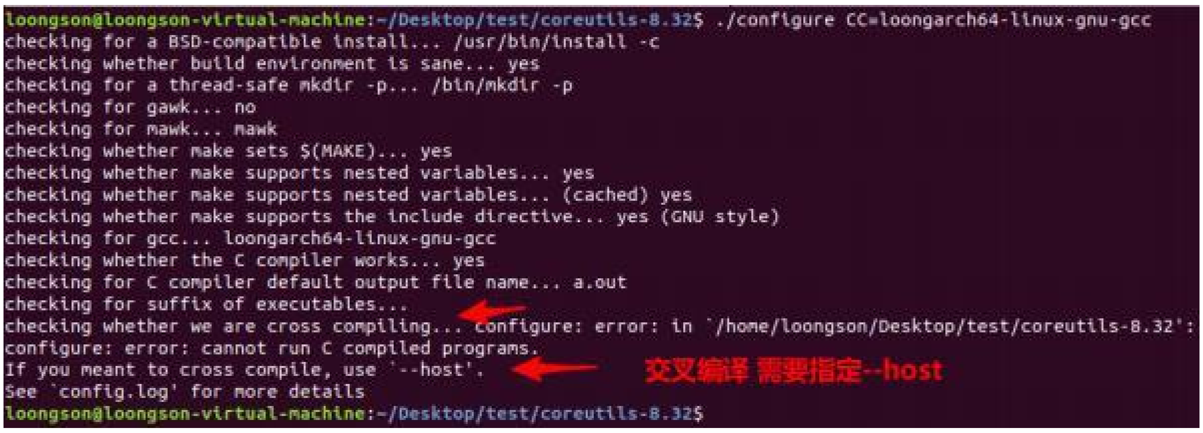
下面将以编译coreutils-8.32为例子说明。

输入./configiure --help的时候会发现CC这个属性，这个就是gcc的值。理论上只要声明工具链，然后执 行configure时指定参数CC= loongarch64 - linux-gnu-gcc的话，就能指定工具链。



```
  CC          C compiler command
  CFLAGS      C compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
              nonstandard directory <lib dir>
  LIBS        libraries to pass to the linker, e.g. -l<library>
  CPPFLAGS    (Objective) C/C++ preprocessor flags, e.g. -I<include dir> if
              you have headers in a nonstandard directory <include dir>
  CPP         C preprocessor
  YACC        The "Yet Another C Compiler" implementation to use. Defaults to
              'bison -o y.tab.c'. Values other than 'bison -o y.tab.c' will
              most likely break on most systems.
  YFLAGS      YFLAGS contains the list arguments that will be passed by
              default to Bison. This script will default YFLAGS to the empty
              string to avoid a default value of '-d' given by some make
              applications.
  DEFAULT_POSIX2_VERSION
              POSIX version to default to; see 'config.hin'.

Use these variables to override the choices made by `configure' or to help
it to find libraries and programs with nonstandard names/locations.
```

但是会发现报以下的错误：



```
loongson@loongson-virtual-machine:~/Desktop/test/coreutils-8.32$ ./configure CC=loongarch64-linux-gnu-gcc
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether make supports nested variables... (cached) yes
checking whether make supports the include directive... yes (GNU style)
checking for gcc... loongarch64-linux-gnu-gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... configure: error: in `/home/loongson/Desktop/test/coreutils-8.32':
configure: error: cannot run C compiled programs.
If you meant to cross compile, use `--host'.
See `config.log' for more details
loongson@loongson-virtual-machine:~/Desktop/test/coreutils-8.32$
```

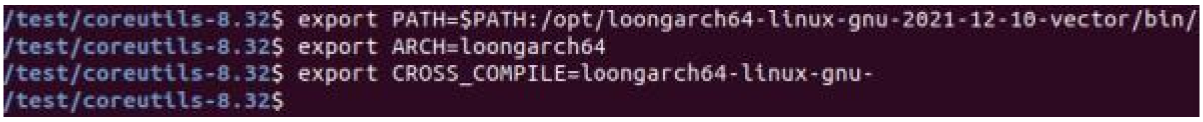
所以建议的操作如下：

声明工具链，即在shell终端输入以下命令：（这个只是例子，按实际情况而定）

export PATH=$PATH:/opt/loongson-gnu-toolchain-8.3-x86_64-

loongarch64-linux-gnu-rc1.3-1/bin/ export ARCH=loongarch64

export CROSS_COMPILE=loongarch64-linux-gnu-

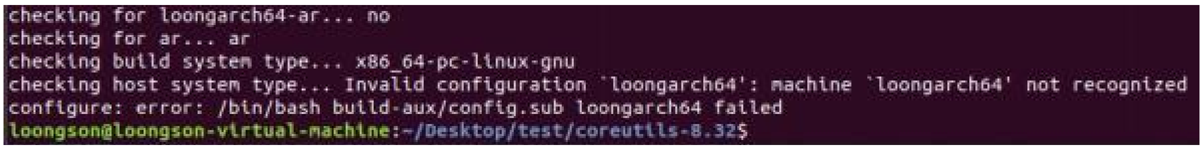


```
/test/coreutils-8.32$ export PATH=$PATH:/opt/loongarch64-linux-gnu-2021-12-10-vector/bin/
/test/coreutils-8.32$ export ARCH=loongarch64
/test/coreutils-8.32$ export CROSS_COMPILE=loongarch64-linux-gnu-
/test/coreutils-8.32$
```

输入命令：

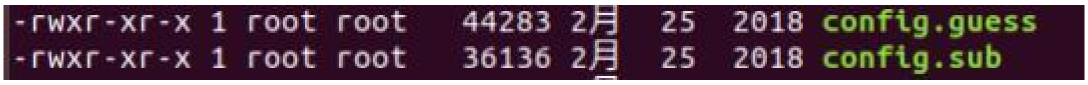
```
./configure --host=loongarch64 CC=loongarch64-linux-gnu-gcc
```

而对于loongarch架构，因为这是新的架构，目前ubuntu18.04里面是没有这个架构的记录的。进而报这 个错误。



```
checking for loongarch64-ar... no
checking for ar... ar
checking build system type... x86_64-pc-linux-gnu
checking host system type... Invalid configuration `loongarch64': machine `loongarch64' not recognized
configure: error: /bin/bash build-aux/config.sub loongarch64 failed
loongson@loongson-virtual-machine:~/Desktop/test/coreutils-8.32$
```

这是在build-aux/config.sub脚本里面没有loongarch64的记录。而在系统的/usr/share/misc文件夹下 面，有以下的文件。



```
-rwxr-xr-x 1 root root   44283 2月   25  2018 config.guess
-rwxr-xr-x 1 root root   36136 2月   25  2018 config.sub
```

那么建议切换到root用户下(su root)，在/usr/share/misc的文件夹下对config.guess和config.sub文件进 行添加（建议事先备份）或者下载最新的config.guess和config.sub 文件。下载方法：

下载config.sub

```
sudo wget -O /usr/share/misc/config.sub
"git.savannah.gnu.org/gitweb/?
p=config.git;a=blob_plain;f=config.sub;hb=HEAD"
```

下载config.guess

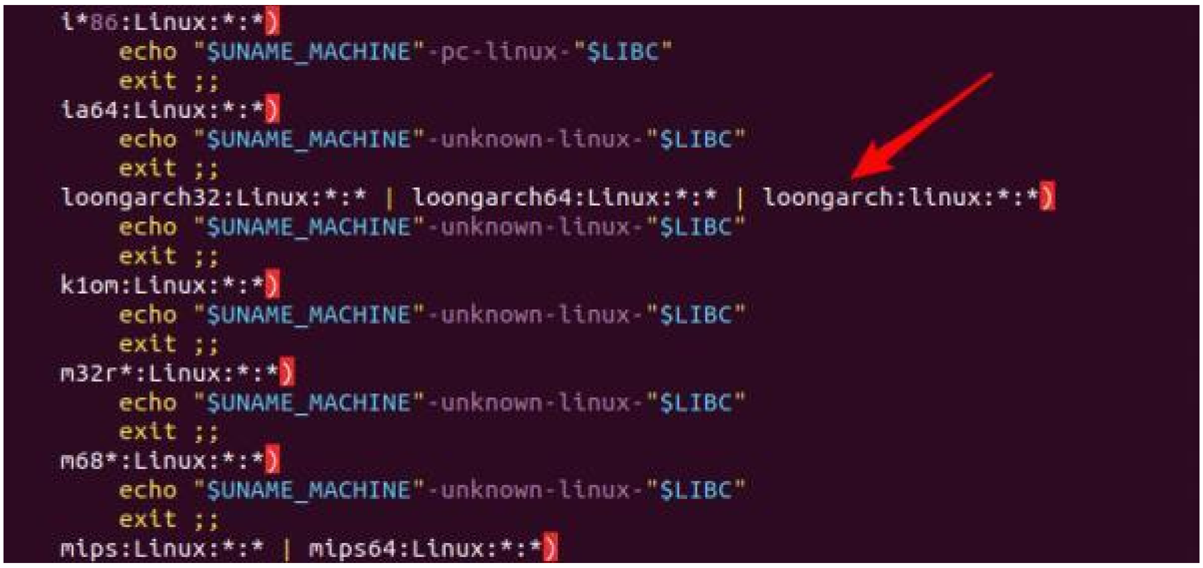
```
sudo wget -O /usr/share/misc/config.guess
"git.savannah.gnu.org/gitweb/?
p=config.git;a=blob_plain;f=config.guess;hb=HEAD"
```

如手动修改config.guess或config.sub则添加内容

如下： config.guess文件的964行附近，添加

```
loongarch32:Linux:\*:\* | loongarch64:Linux:\*:\* | loongarch:linux:\*:\*)
        echo "$UNAME_MACHINE"-unknown-linux-
        "$LIBC" exit ;;
```

如下图：

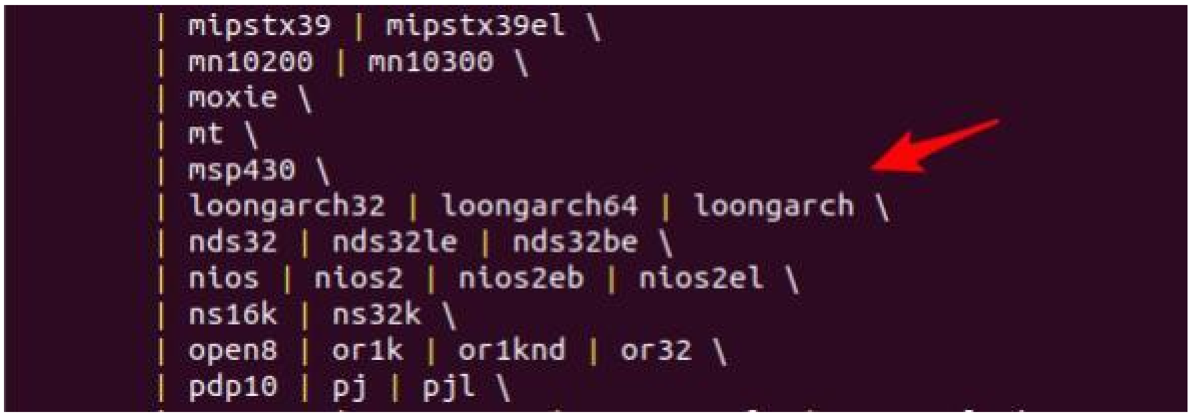


```
    i*86:Linux:*:*)
        echo "$UNAME_MACHINE"-pc-linux-"$LIBC"
        exit ;;
    ia64:Linux:*:*)
        echo "$UNAME_MACHINE"-unknown-linux-"$LIBC"
        exit ;;
    loongarch32:Linux:*:* | loongarch64:Linux:*:* | loongarch:linux:*:*)
        echo "$UNAME_MACHINE"-unknown-linux-"$LIBC"
        exit ;;
    k1om:Linux:*:*)
        echo "$UNAME_MACHINE"-unknown-linux-"$LIBC"
        exit ;;
    m32r*:Linux:*:*)
        echo "$UNAME_MACHINE"-unknown-linux-"$LIBC"
        exit ;;
    m68*:Linux:*:*)
        echo "$UNAME_MACHINE"-unknown-linux-"$LIBC"
        exit ;;
    mips:Linux:*:* | mips64:Linux:*:*)
```

在config.sub文件的295行附近，添加：

```
| loongarch32 | loongarch64 | loongarch \
```

如下图：

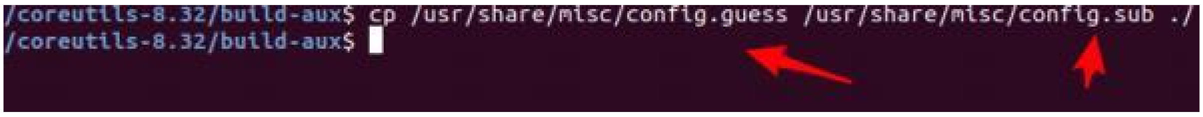


```
        | mipstx39 | mipstx39el \
        | mn10200 | mn10300 \
        | moxie \
        | mt \
        | msp430 \
        | loongarch32 | loongarch64 | loongarch \
        | nds32 | nds32le | nds32be \
        | nios | nios2 | nios2eb | nios2el \
        | ns16k | ns32k \
        | open8 | or1k | or1knd | or32 \
        | pdp10 | pj | pjl \
```

可见添加内容都是仿照其他架构的内容。添加的位置不是指定的。只需要模仿得当即可。

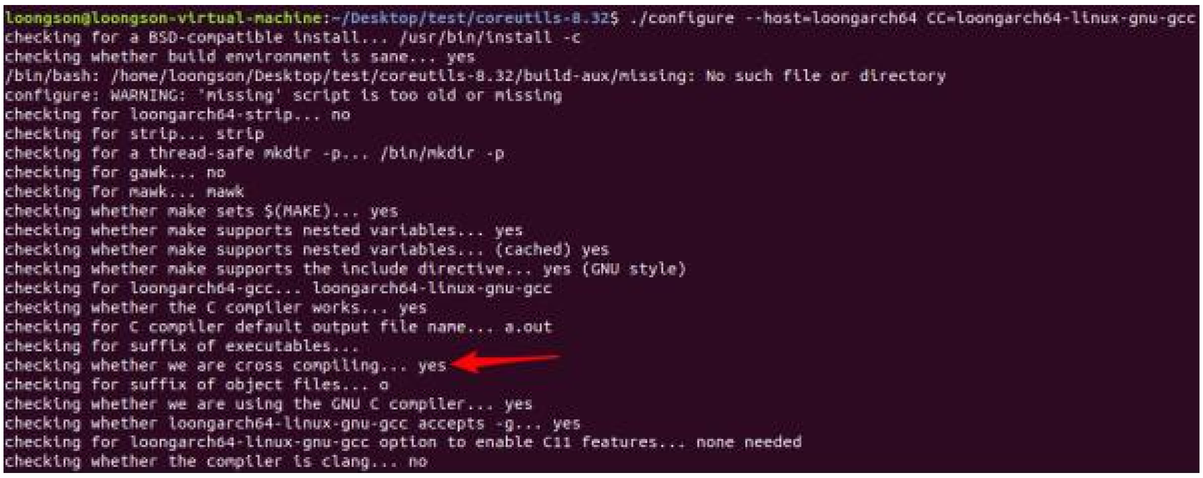
然后在build-aux里面执行以下命令（**注意不是root用户**）：

```
cp /usr/share/misc/config.guess /usr/share/misc/config.sub ./
```

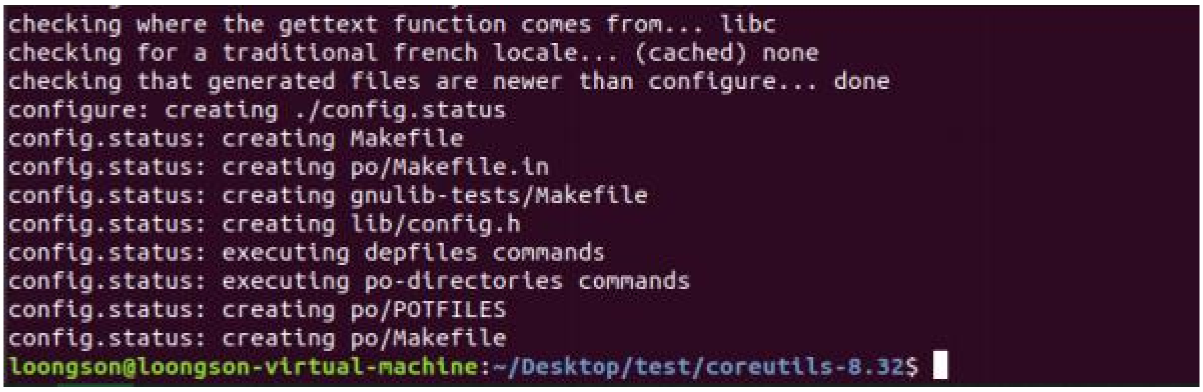


```
/coreutils-8.32/build-aux$ cp /usr/share/misc/config.guess /usr/share/misc/config.sub ./
/coreutils-8.32/build-aux$
```

然后configure就能成功执行。



```
loongson@loongson-virtual-machine:~/Desktop/test/coreutils-8.32$ ./configure --host=loongarch64 CC=loongarch64-linux-gnu-gcc
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
/bin/bash: /home/loongson/Desktop/test/coreutils-8.32/build-aux/missing: No such file or directory
configure: WARNING: 'missing' script is too old or missing
checking for loongarch64-strip... no
checking for strip... strip
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether make supports nested variables... (cached) yes
checking whether make supports the include directive... yes (GNU style)
checking for loongarch64-gcc... loongarch64-linux-gnu-gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... yes
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether loongarch64-linux-gnu-gcc accepts -g... yes
checking for loongarch64-linux-gnu-gcc option to enable C11 features... none needed
checking whether the compiler is clang... no
```

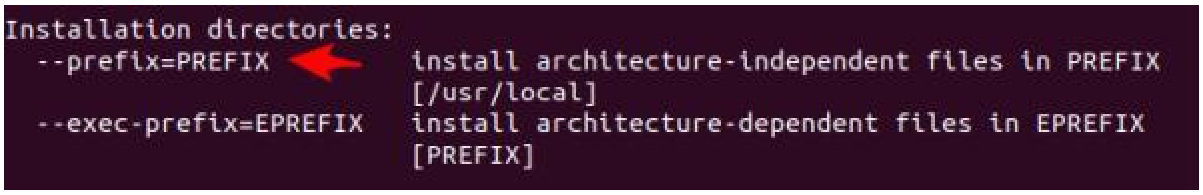


```
checking where the gettext function comes from... libc
checking for a traditional french locale... (cached) none
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating po/Makefile.in
config.status: creating gnulib-tests/Makefile
config.status: creating lib/config.h
config.status: executing depfiles commands
config.status: executing po-directories commands
config.status: creating po/POTFILES
config.status: creating po/Makefile
loongson@loongson-virtual-machine:~/Desktop/test/coreutils-8.32$
```

随后只需要make即可编译完成。

注意交叉编译的产物不是编译机能用的程序，所以make install的话，就会把不属于编译机的程序安装 到编译机的/usr中。那么将会是比较麻烦的事情。

所以请提前准备好安装的目录，并且在configure的参数中声明或者make install的时候指定安装路径。 通常--prefix指的是安装路径。建议在configure的时候指定好。



```
Installation directories:
  --prefix=PREFIX         install architecture-independent files in PREFIX
                          [/usr/local]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
                          [PREFIX]
```

其他更加精细的安装目录设置则根据实际情况与configure的特性而自行指定。

对于其他源码包，可能关于config.guess的报错不只一处，还请按照实际情况做出对应操作。

8.6.Qt交叉编译软件

8.6.1.直接利用buildroot编译完成的QT交叉工具链

buildroot 编译完成后会生成QT交叉编译工具链，可以直接利用（参考9.1.1.buildroot编译）：

```
# cd buildroot-2021.02/qt_test #
source env_loongarch64.sh
```

env_loongarch64.sh 的内容如下：

```
workdir=`pwd`
DEST=$workdir/../output/host
PATH=$DEST/bin:$PATH
QMAKESPEC=$DEST/mkspecs/linux-loongarch64-g++/
QTDIR=$DEST/loongarch64-buildroot-linux-gnu/sysroot/usr
LD_LIBRARY_PATH=$DEST/loongarch64-buildroot-linux-gnu/sysroot/usr/lib

export PATH QMAKESPEC QTDIR LD_LIBARY_PATH
```

此时在应用代码下执行 qmake 会使用buildroot编译生成的QT工具链：

```
# qmke
Info: creating stash file /home/loongson/buildroot-2021.02/qt_test/.qmake.stash # make
```

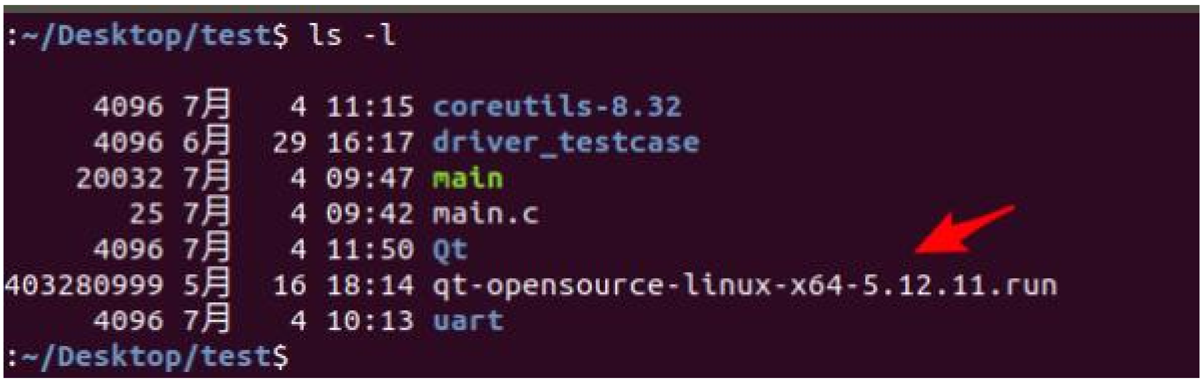
最后生成可执行文件就是 loongarch 的二进制文件。

8.6.2.ubuntu下从头部署QT交叉编译工具

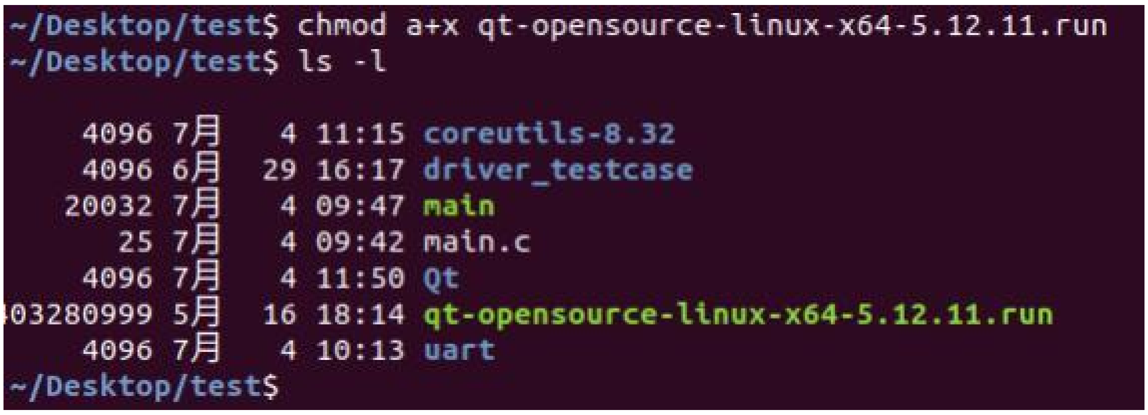
我们也可以不用buildroot编译出的工具链，从头部署：

> 注意本文演示的系统是ubuntu18.04系统，如果是开发板资料中提供的软件包，那么不需要管Qt的 部署，因为已经设置好了，直接前往编译就好。

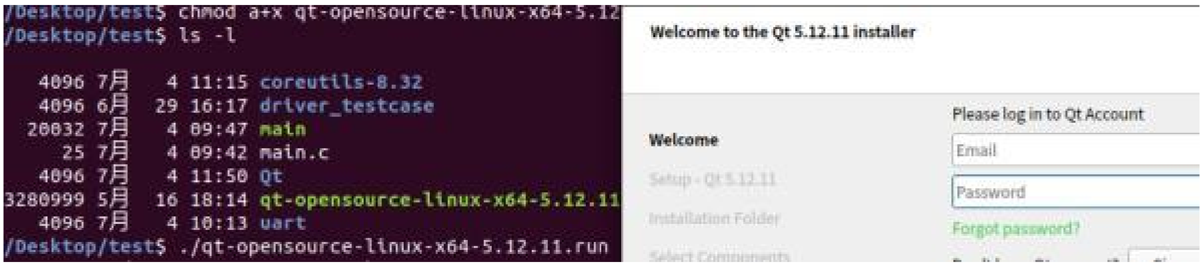
首先， ubuntu18.04需要安装好Qt环境和QtCreator。开发板资料中有一个qt-opensource-linux-x64- 5.12.11.run包，可以在ubuntu18.04里面直接运行，即可安装上述的两个条件。



```
:~/Desktop/test$ ls -l

     4096 7月    4 11:15 coreutils-8.32
     4096 6月   29 16:17 driver_testcase
    20032 7月    4 09:47 main
       25 7月    4 09:42 main.c
     4096 7月    4 11:50 Qt
403280999 5月   16 18:14 qt-opensource-linux-x64-5.12.11.run
     4096 7月    4 10:13 uart
:~/Desktop/test$
```



```
~/Desktop/test$ chmod a+x qt-opensource-linux-x64-5.12.11.run
~/Desktop/test$ ls -l

     4096 7月    4 11:15 coreutils-8.32
     4096 6月   29 16:17 driver_testcase
    20032 7月    4 09:47 main
       25 7月    4 09:42 main.c
     4096 7月    4 11:50 Qt
03280999 5月   16 18:14 qt-opensource-linux-x64-5.12.11.run
     4096 7月    4 10:13 uart
~/Desktop/test$
```



```
/Desktop/test$ chmod a+x qt-opensource-linux-x64-5.12
/Desktop/test$ ls -l

    4096 7月    4 11:15 coreutils-8.32
    4096 6月   29 16:17 driver_testcase
   20032 7月    4 09:47 main
      25 7月    4 09:42 main.c
    4096 7月    4 11:50 Qt
3280999 5月   16 18:14 qt-opensource-linux-x64-5.12.11
    4096 7月    4 10:13 uart
/Desktop/test$ ./qt-opensource-linux-x64-5.12.11.run
```

Welcome to the Qt 5.12.11 installer

随后根据安装提示安装即可。

注意上面只是推荐了一种开发方式。即Qt的UI界面和一些逻辑处理可以在ubuntu18.04里面验证，然后 再使用交叉编译，到板卡上运行验证。

下面分以下步骤说明：部署loongarch64版本的Qt套件，部署Qtcreator的交叉编译配置。选择编译配 置。

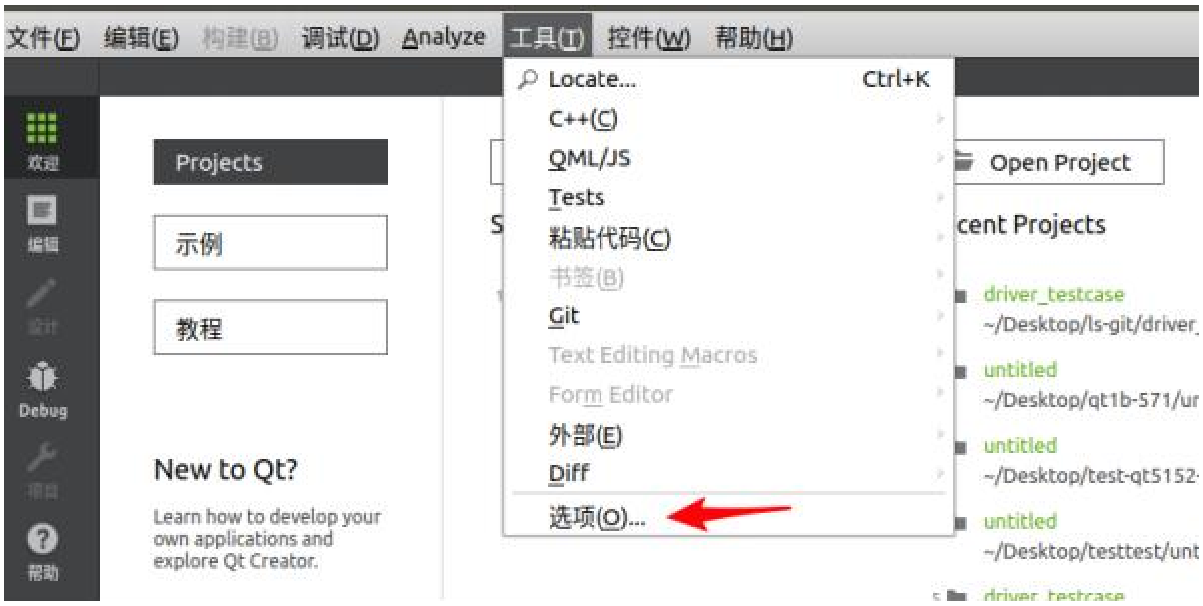
先部署loongarch64版本的Qt套件，在开发板资料中有Qt-5.15.2-LA64.tar.gz的压缩包。请复制到 ubuntu18.04的里面(路径不做要求)。然后在压缩包所在的文件夹中打开终端，输入：

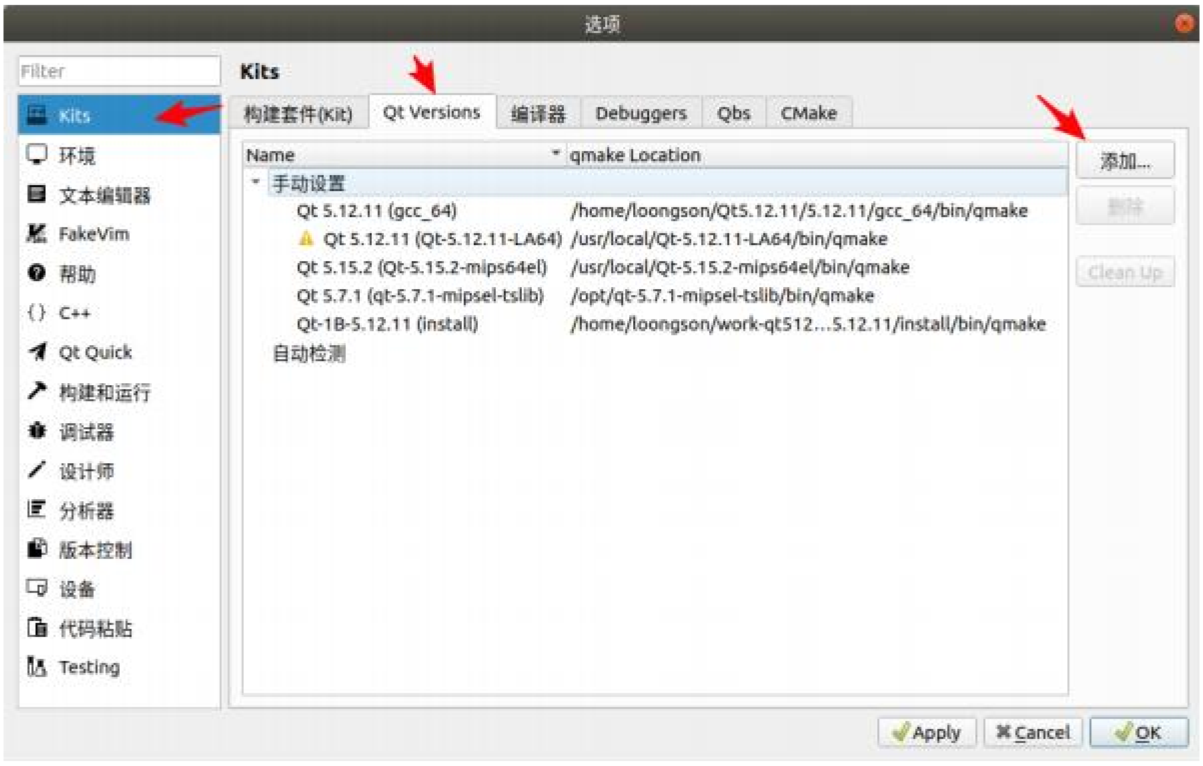
```
sudo tar -zxf Qt-5.15.2-LA64.tar.gz -C /usr/local
```

那么/usr/local/下就会存在文件夹Qt-5.15.2-LA64。那么部署成功。

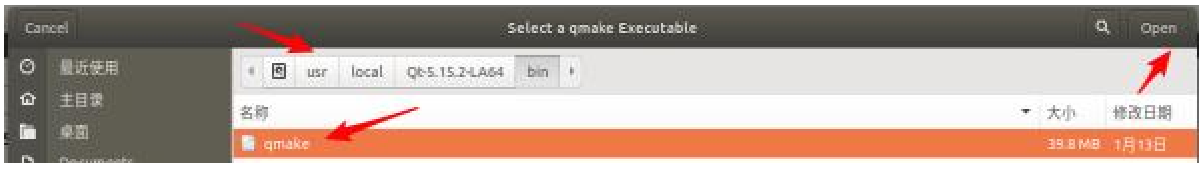
然后就是Qtcreator的部署。例子如下：

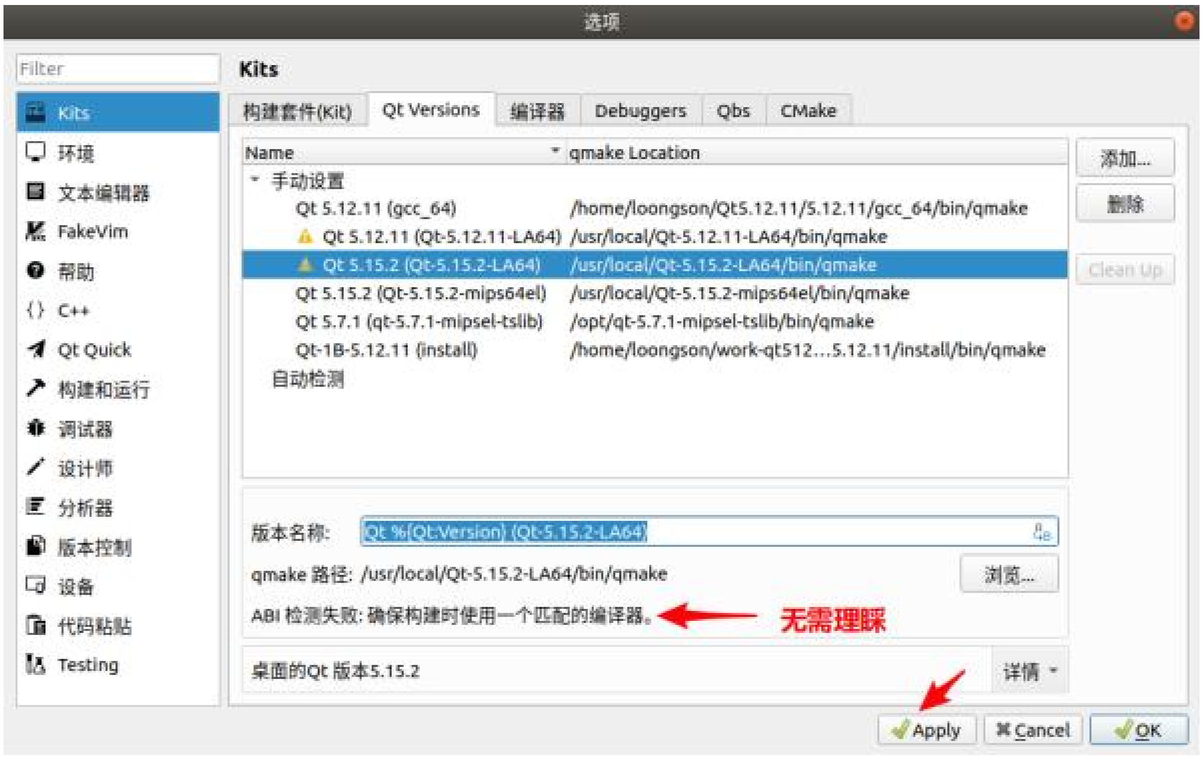
打开Qtcreator，按下面的图示操作执行。





Name	qmake Location
手动设置	
Qt 5.12.11 (gcc_64)	/home/loongson/Qt5.12.11/5.12.11/gcc_64/bin/qmake
⚠ Qt 5.12.11 (Qt-5.12.11-LA64)	/usr/local/Qt-5.12.11-LA64/bin/qmake
Qt 5.15.2 (Qt-5.15.2-mips64el)	/usr/local/Qt-5.15.2-mips64el/bin/qmake
Qt 5.7.1 (qt-5.7.1-mipsel-tslib)	/opt/qt-5.7.1-mipsel-tslib/bin/qmake
Qt-1B-5.12.11 (install)	/home/loongson/work-qt512...5.12.11/install/bin/qmake
自动检测	





Name	qmake Location
手动设置	
Qt 5.12.11 (gcc_64)	/home/loongson/Qt5.12.11/5.12.11/gcc_64/bin/qmake
⚠ Qt 5.12.11 (Qt-5.12.11-LA64)	/usr/local/Qt-5.12.11-LA64/bin/qmake
⚠ Qt 5.15.2 (Qt-5.15.2-LA64)	/usr/local/Qt-5.15.2-LA64/bin/qmake
Qt 5.15.2 (Qt-5.15.2-mips64el)	/usr/local/Qt-5.15.2-mips64el/bin/qmake
Qt 5.7.1 (qt-5.7.1-mipsel-tslib)	/opt/qt-5.7.1-mipsel-tslib/bin/qmake
Qt-1B-5.12.11 (install)	/home/loongson/work-qt512...5.12.11/install/bin/qmake
自动检测	

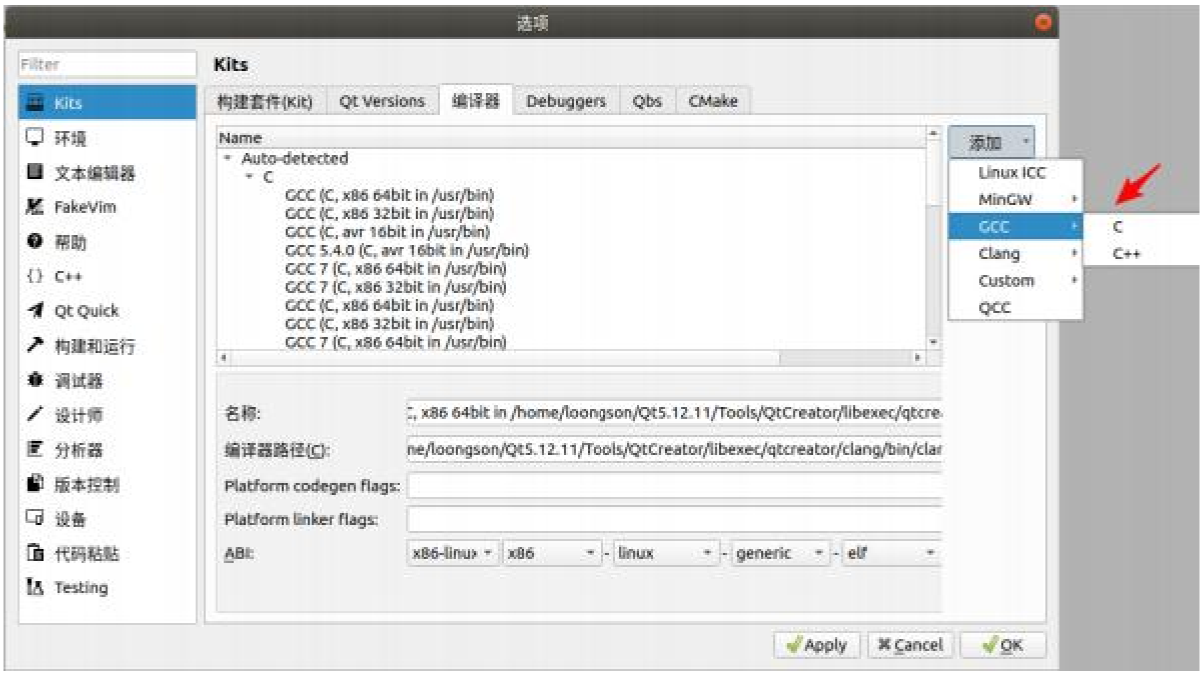
版本名称: Qt %{Qt:Version} (Qt-5.15.2-LA64)

qmake 路径: /usr/local/Qt-5.15.2-LA64/bin/qmake

ABI 检测失败: 确保构建时使用一个匹配的编译器。

无需理睬

桌面的Qt 版本5.15.2



```
Name
Auto-detected
  C
      GCC (C, x86 64bit in /usr/bin)
      GCC (C, x86 32bit in /usr/bin)
      GCC (C, avr 16bit in /usr/bin)
      GCC 5.4.0 (C, avr 16bit in /usr/bin)
      GCC 7 (C, x86 64bit in /usr/bin)
      GCC 7 (C, x86 32bit in /usr/bin)
      GCC (C, x86 64bit in /usr/bin)
      GCC (C, x86 32bit in /usr/bin)
      GCC 7 (C, x86 64bit in /usr/bin)
```

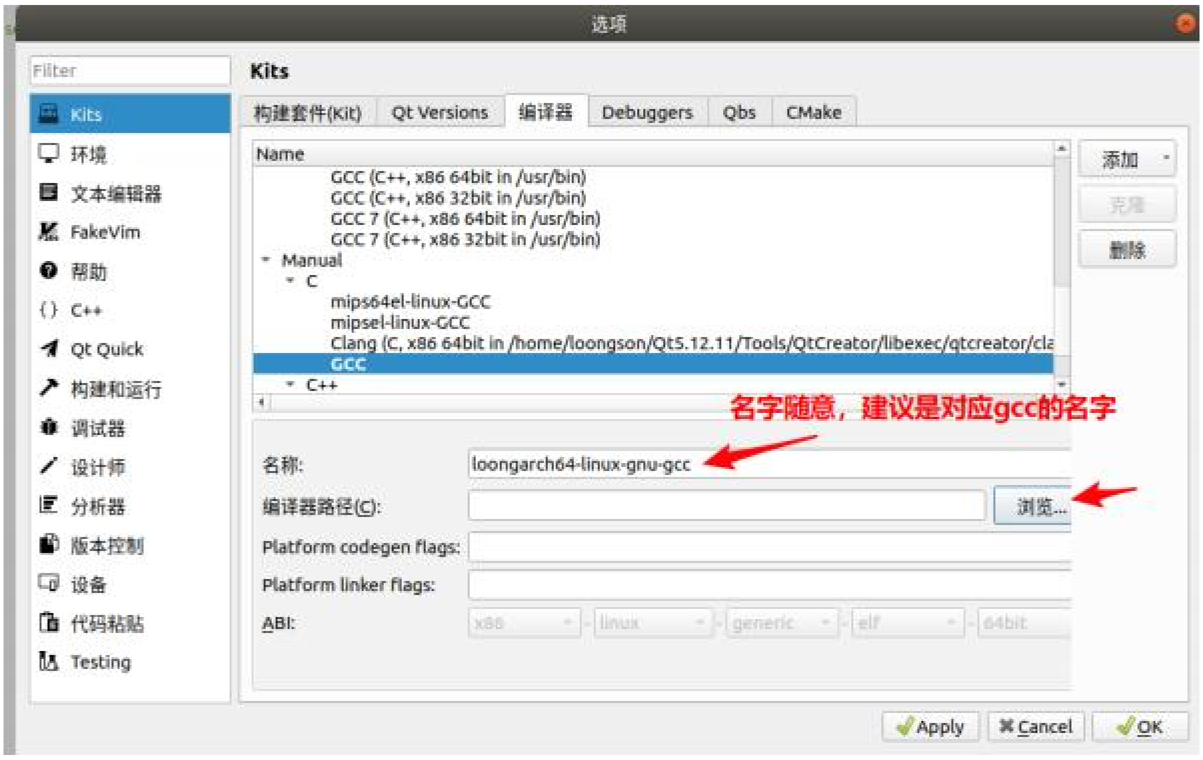
名称: C, x86 64bit in /home/loongson/Qt5.12.11/Tools/QtCreator/libexec/qtcre

编译器路径(C): ne/loongson/Qt5.12.11/Tools/QtCreator/libexec/qtcreator/clang/bin/clar

Platform codegen flags:

Platform linker flags:

ABI: x86-linux - x86 - linux - generic - elf


选项

Filter

Kits

环境
文本编辑器
FakeVim
帮助
C++
Qt Quick
构建和运行
调试器
设计师
分析器
版本控制
设备
代码粘贴
Testing

构建套件(Kit) | Qt Versions | 编译器 | Debuggers | Qbs | CMake

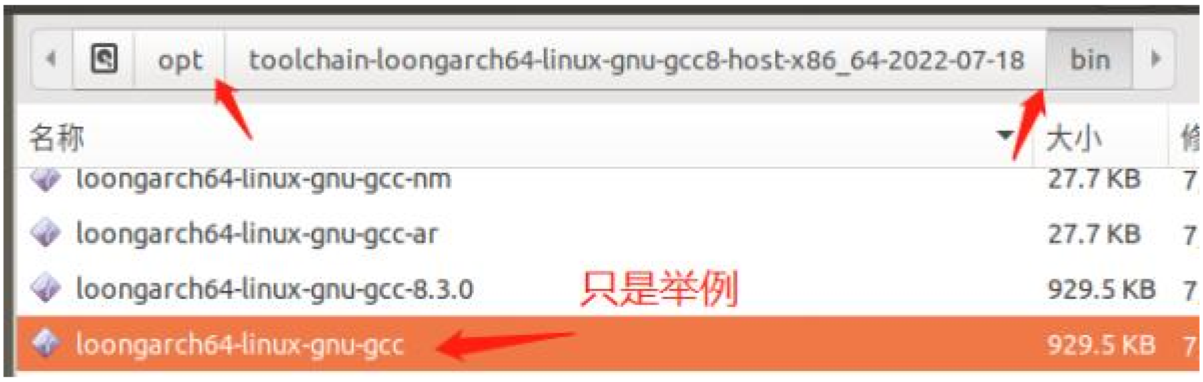
Name
GCC (C++, x86 64bit in /usr/bin)
GCC (C++, x86 32bit in /usr/bin)
GCC 7 (C++, x86 64bit in /usr/bin)
GCC 7 (C++, x86 32bit in /usr/bin)
Manual
C
mips64el-linux-GCC
mipsel-linux-GCC
Clang (C, x86 64bit in /home/loongson/Qt5.12.11/Tools/QtCreator/libexec/qtcreator/cla
GCC
C++

添加
克隆
删除

名字随意，建议是对应gcc的名字

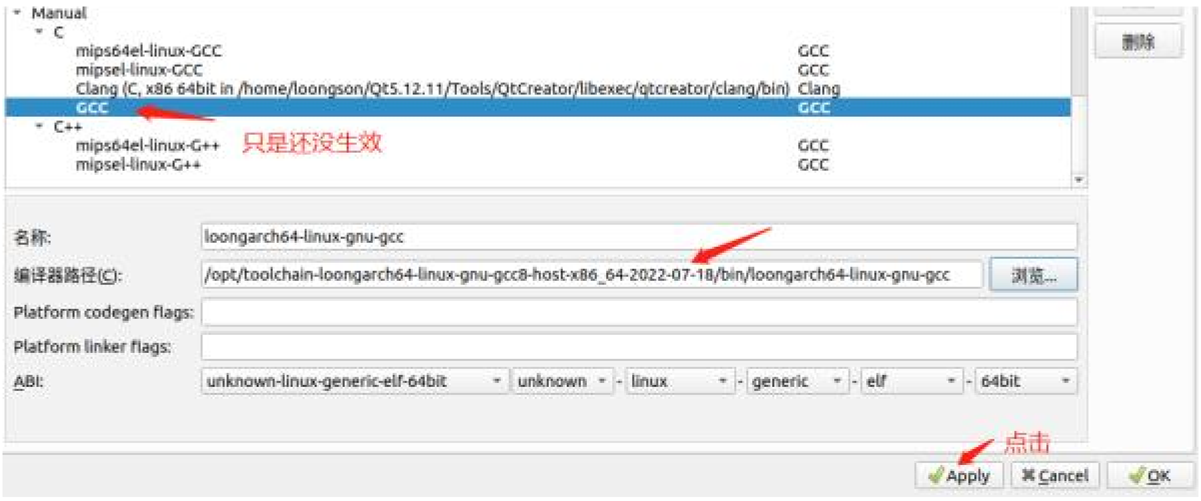
名称: loongarch64-linux-gnu-gcc
编译器路径(C): 浏览...
Platform codegen flags:
Platform linker flags:
ABI: x86 - linux - generic - elf - 64bit

Apply Cancel OK


opt | toolchain-loongarch64-linux-gnu-gcc8-host-x86_64-2022-07-18 | bin

名称	大小
loongarch64-linux-gnu-gcc-nm	27.7 KB
loongarch64-linux-gnu-gcc-ar	27.7 KB
loongarch64-linux-gnu-gcc-8.3.0	929.5 KB
loongarch64-linux-gnu-gcc	929.5 KB

只是举例


Manual
C
mips64el-linux-GCC   GCC
mipsel-linux-GCC   GCC
Clang (C, x86 64bit in /home/loongson/Qt5.12.11/Tools/QtCreator/libexec/qtcreator/clang/bin)   Clang
GCC   GCC
C++
mips64el-linux-G++   GCC
mipsel-linux-G++   GCC

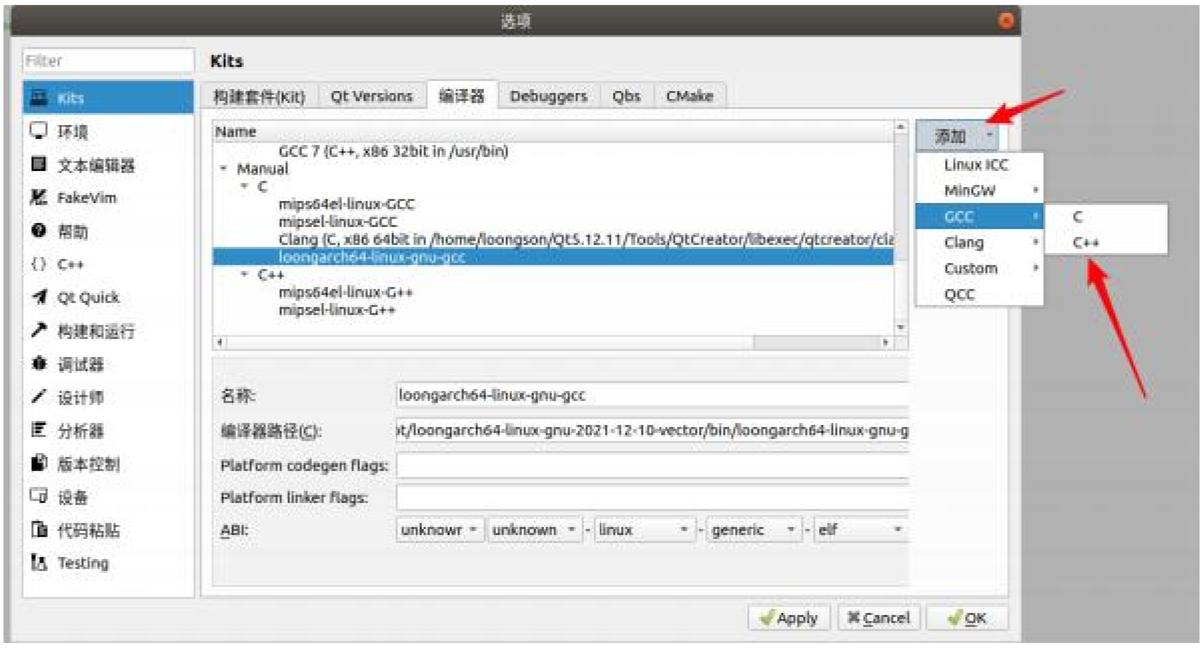
只是还没生效

删除

名称: loongarch64-linux-gnu-gcc
编译器路径(C): /opt/toolchain-loongarch64-linux-gnu-gcc8-host-x86_64-2022-07-18/bin/loongarch64-linux-gnu-gcc 浏览...
Platform codegen flags:
Platform linker flags:
ABI: unknown-linux-generic-elf-64bit | unknown - linux - generic - elf - 64bit

点击

Apply Cancel OK



Kits

Name
- GCC 7 (C++, x86 32bit in /usr/bin)
- Manual
  - C
    - mips64el-linux-GCC
    - mipsel-linux-GCC
    - Clang (C, x86 64bit in /home/loongson/Qt5.12.11/Tools/QtCreator/libexec/qtcreator/cla
    - loongarch64-linux-gnu-gcc
  - C++
    - mips64el-linux-G++
    - mipsel-linux-G++

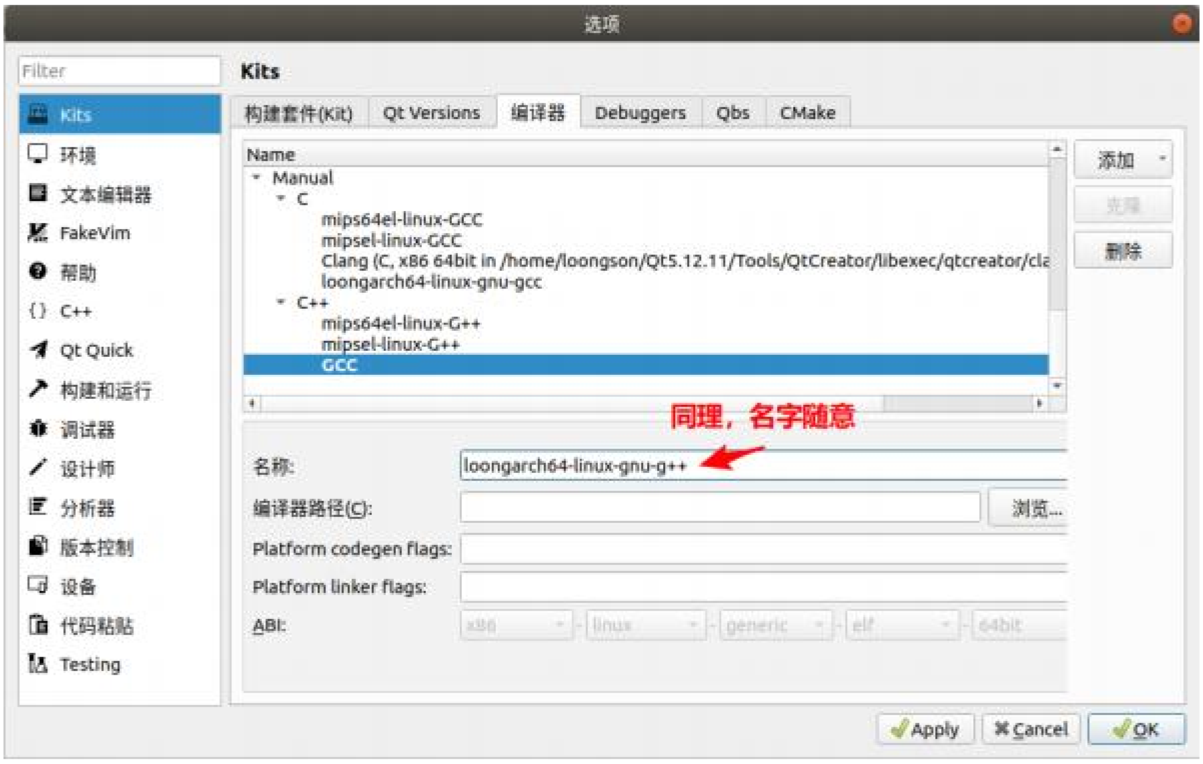
名称: loongarch64-linux-gnu-gcc

编译器路径(C): ɔt/loongarch64-linux-gnu-2021-12-10-vector/bin/loongarch64-linux-gnu-g

Platform codegen flags:

Platform linker flags:

ABI: unknowr - unknown - linux - generic - elf



Kits

Name
- Manual
  - C
    - mips64el-linux-GCC
    - mipsel-linux-GCC
    - Clang (C, x86 64bit in /home/loongson/Qt5.12.11/Tools/QtCreator/libexec/qtcreator/cla
    - loongarch64-linux-gnu-gcc
  - C++
    - mips64el-linux-G++
    - mipsel-linux-G++
    - GCC

同理，名字随意

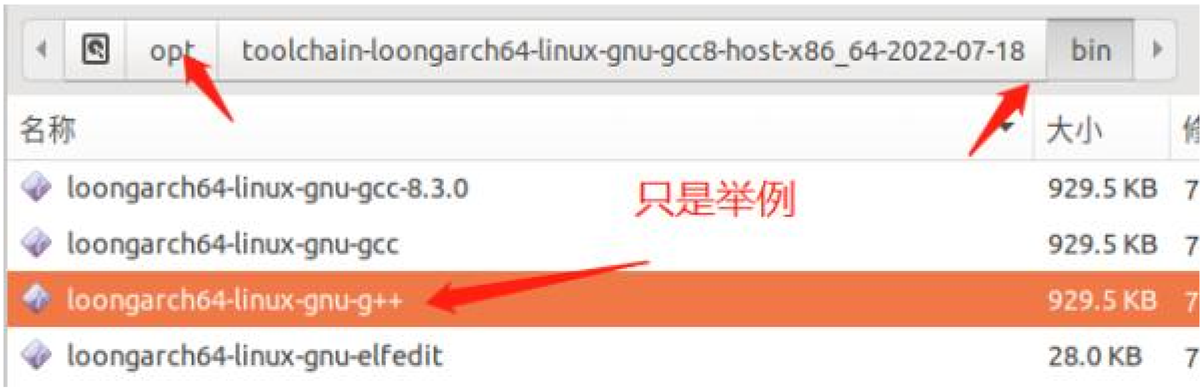
名称: loongarch64-linux-gnu-g++

编译器路径(C):

Platform codegen flags:

Platform linker flags:

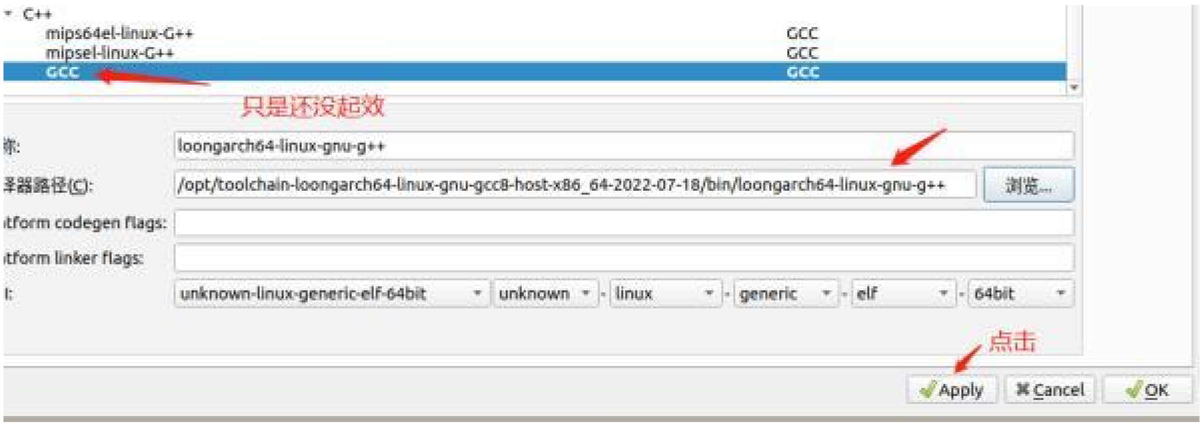
ABI: x86 - linux - generic - elf - 64bit



opt / toolchain-loongarch64-linux-gnu-gcc8-host-x86_64-2022-07-18 / bin

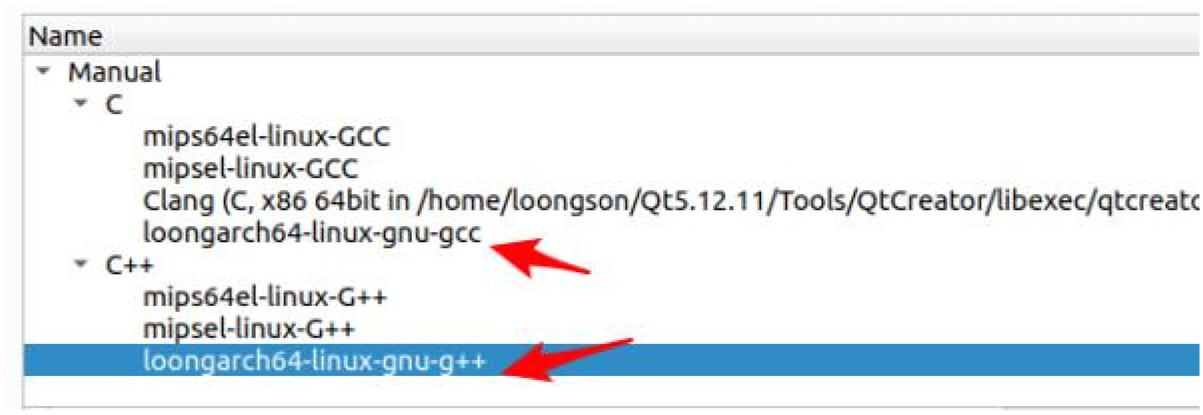
只是举例

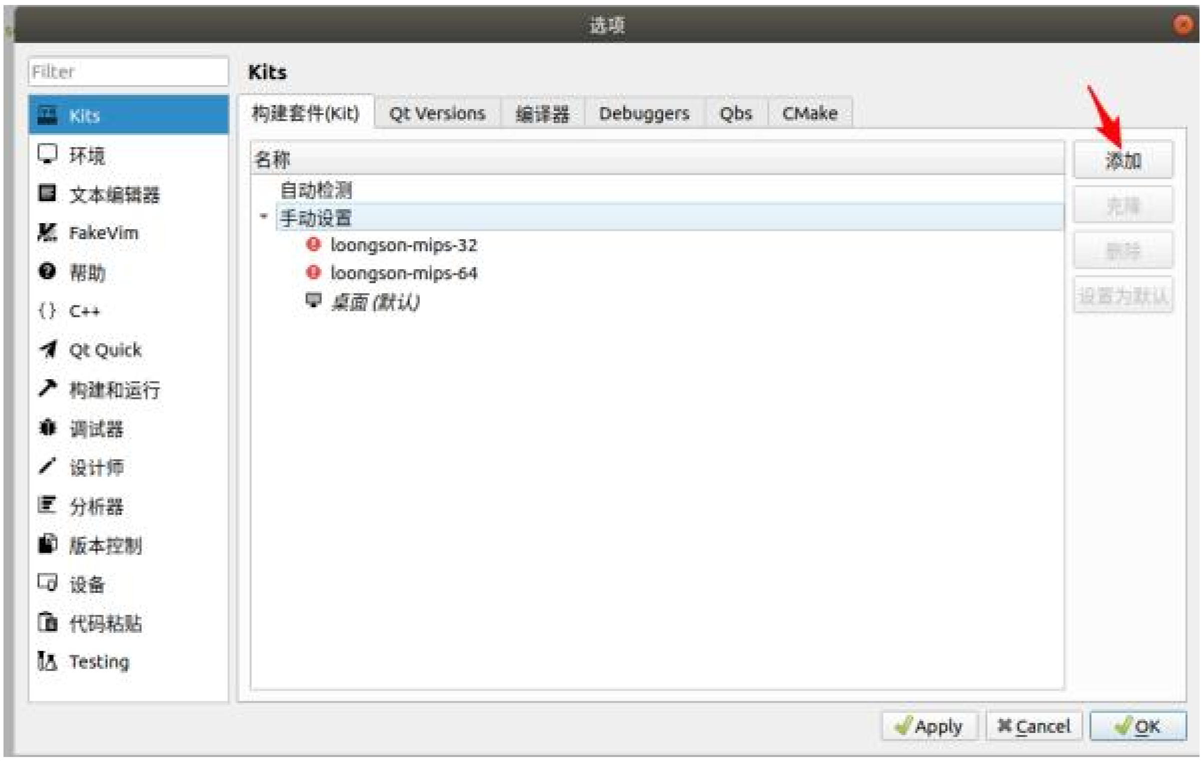
名称	大小
loongarch64-linux-gnu-gcc-8.3.0	929.5 KB
loongarch64-linux-gnu-gcc	929.5 KB
loongarch64-linux-gnu-g++	929.5 KB
loongarch64-linux-gnu-elfedit	28.0 KB


```
▾ C++
    mips64el-linux-G++          GCC
    mipsel-linux-G++            GCC
    GCC                         GCC
```

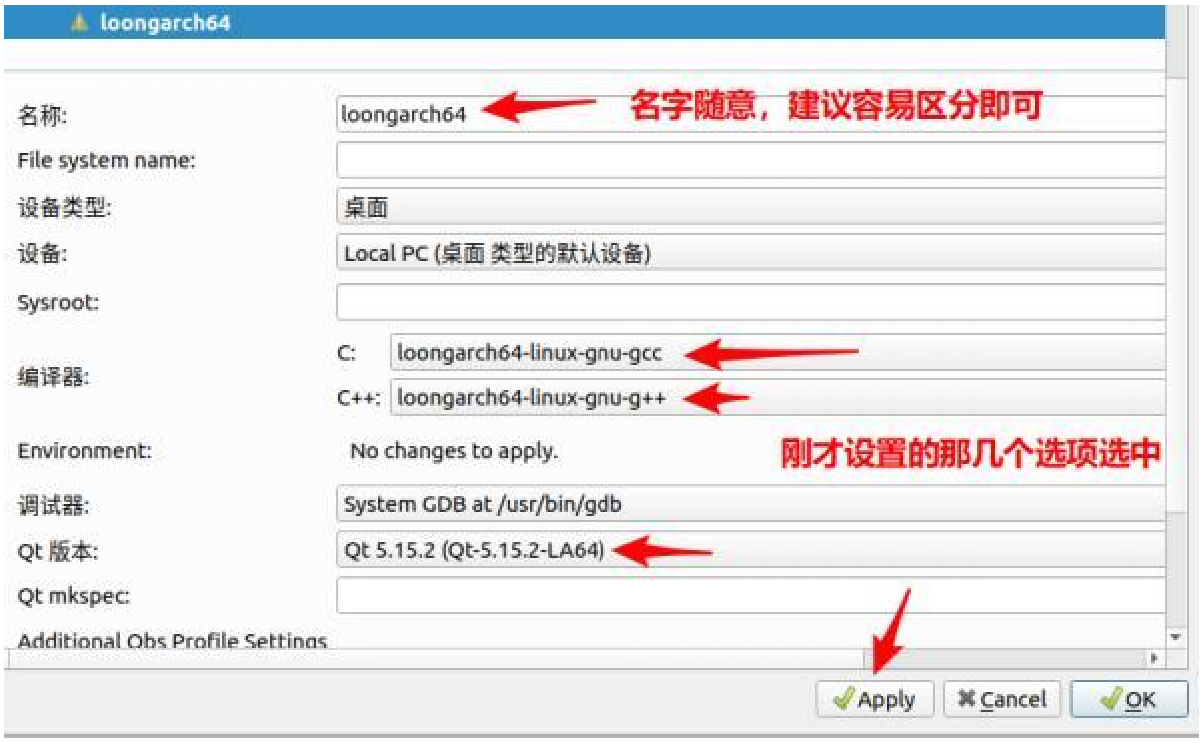
```
称:                   loongarch64-linux-gnu-g++
译器路径(C):          /opt/toolchain-loongarch64-linux-gnu-gcc8-host-x86_64-2022-07-18/bin/loongarch64-linux-gnu-g++   浏览...
tform codegen flags:
tform linker flags:
I:                    unknown-linux-generic-elf-64bit | unknown - linux - generic - elf - 64bit
```

然后就能获得下图所示的内容：


```
Name
▾ Manual
  ▾ C
      mips64el-linux-GCC
      mipsel-linux-GCC
      Clang (C, x86 64bit in /home/loongson/Qt5.12.11/Tools/QtCreator/libexec/qtcreato
      loongarch64-linux-gnu-gcc
  ▾ C++
      mips64el-linux-G++
      mipsel-linux-G++
      loongarch64-linux-gnu-g++
```



```
名称
  自动检测
▾ 手动设置
    loongson-mips-32
    loongson-mips-64
    桌面 (默认)
```

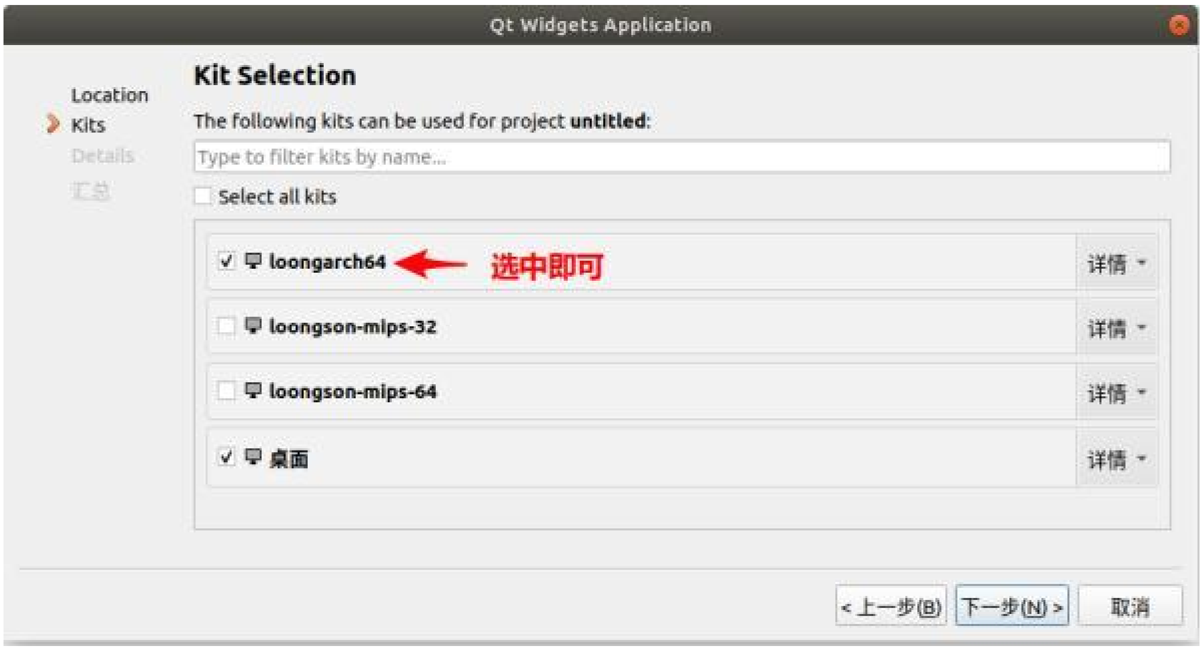


loongarch64

名称:	loongarch64
File system name:	
设备类型:	桌面
设备:	Local PC (桌面 类型的默认设备)
Sysroot:	
编译器:	C: loongarch64-linux-gnu-gcc
	C++: loongarch64-linux-gnu-g++
Environment:	No changes to apply.
调试器:	System GDB at /usr/bin/gdb
Qt 版本:	Qt 5.15.2 (Qt-5.15.2-LA64)
Qt mkspec:	
Additional Qbs Profile Settings	

那么*Qtcreator*的部署就完成了。

如何选择编译配置，如果你是新建工程那么可以参考下图：



Qt Widgets Application

Kit Selection

The following kits can be used for project **untitled**:

Type to filter kits by name...

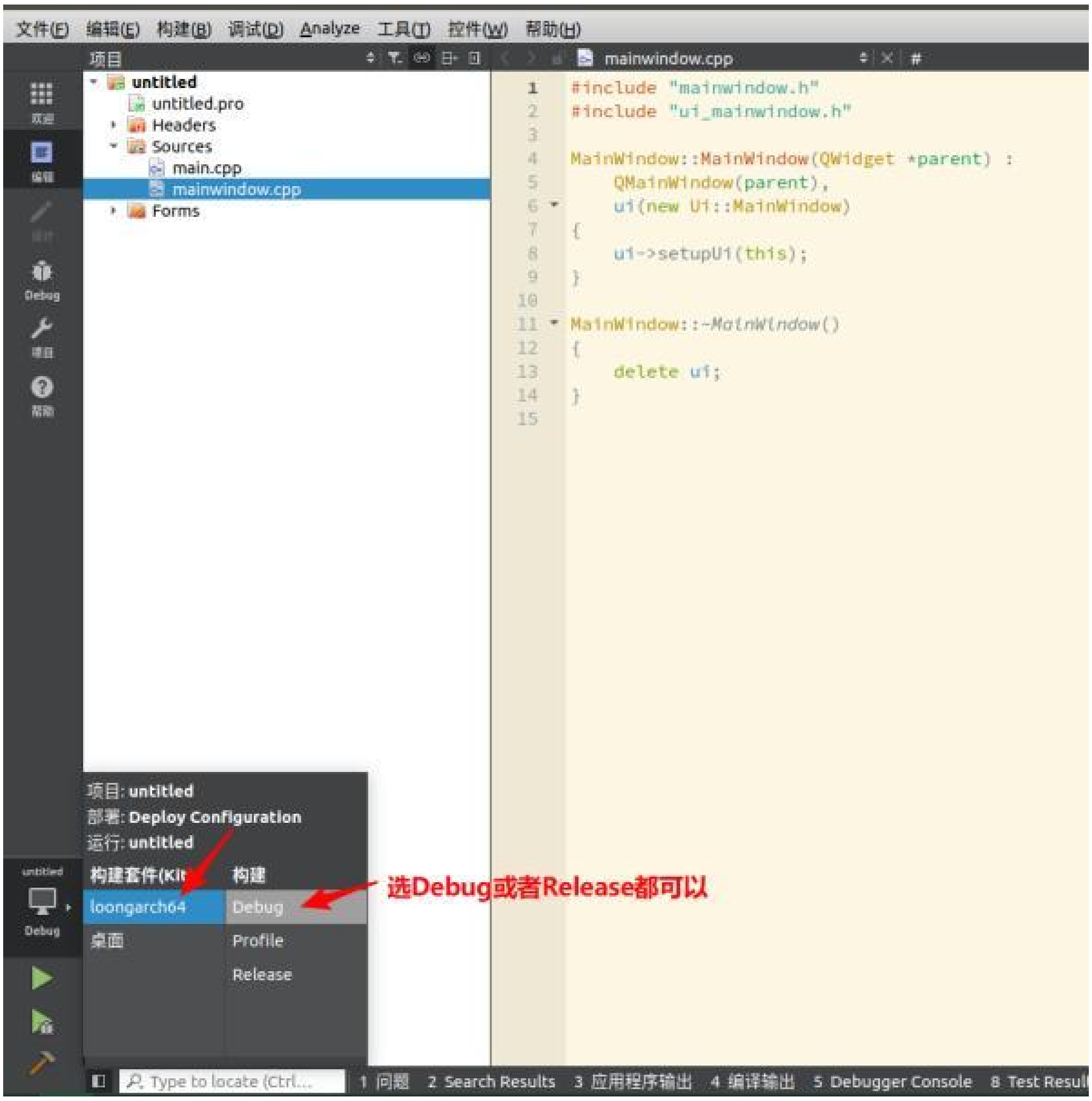
☐ Select all kits

✓ loongarch64

☐ loongson-mips-32

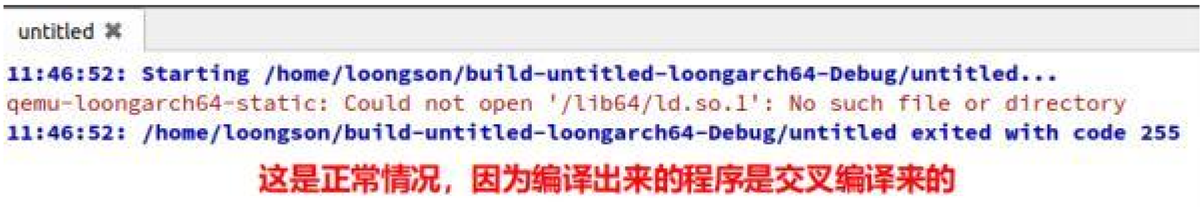
☐ loongson-mips-64

✓ 桌面



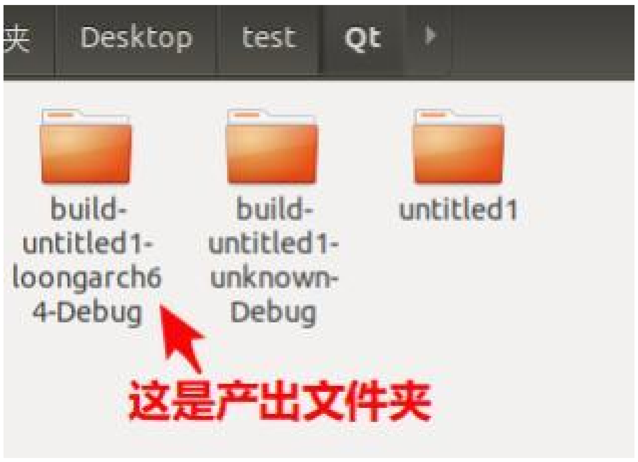
```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

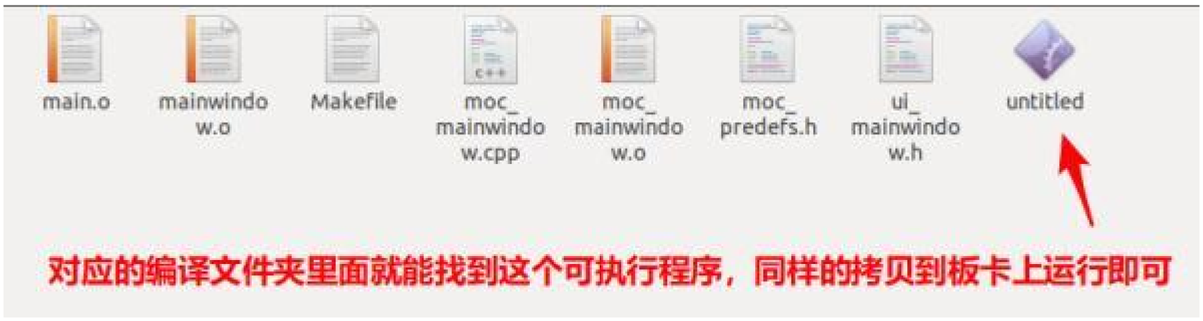
然后照常编译（点击绿色那个三角形）。



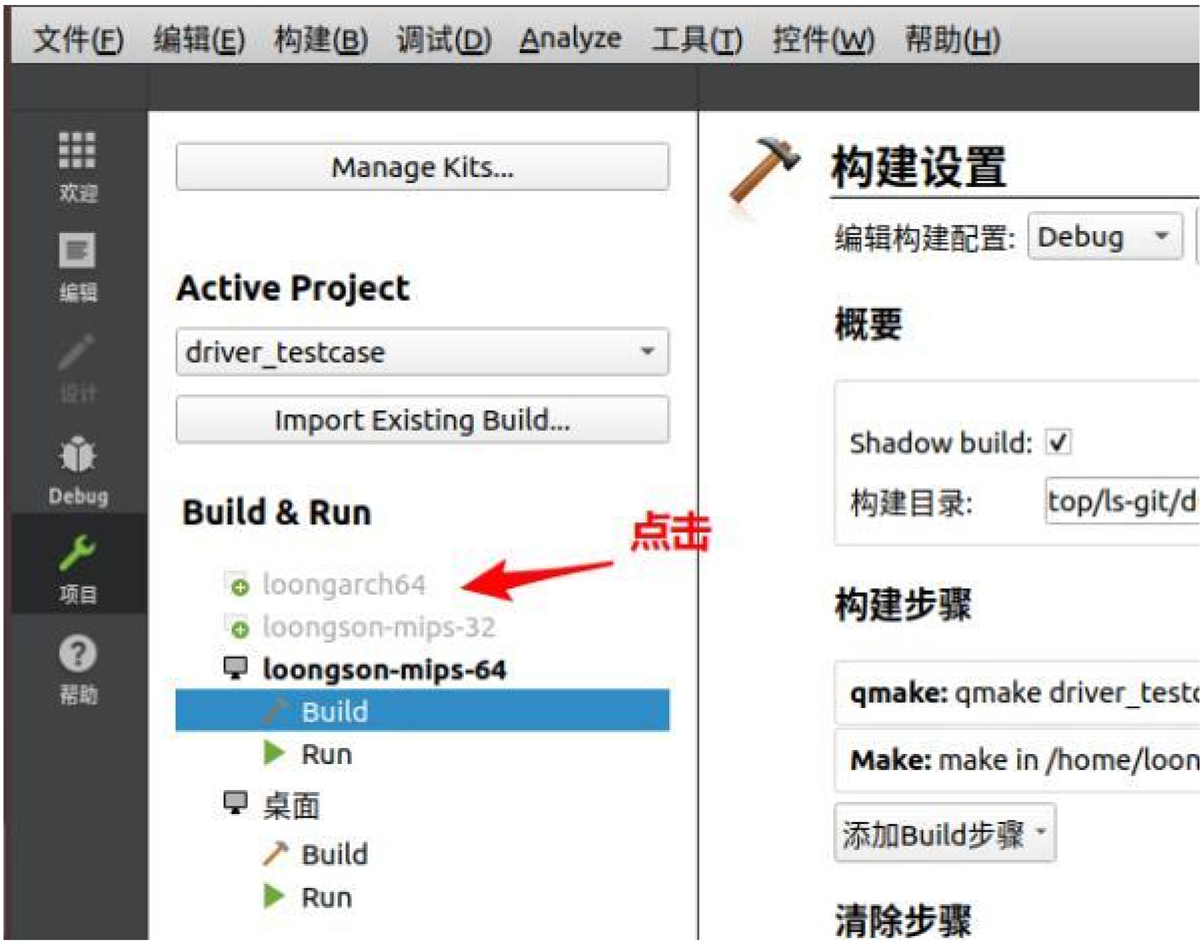
```
11:46:52: Starting /home/loongson/build-untitled-loongarch64-Debug/untitled...
qemu-loongarch64-static: Could not open '/lib64/ld.so.1': No such file or directory
11:46:52: /home/loongson/build-untitled-loongarch64-Debug/untitled exited with code 255
```

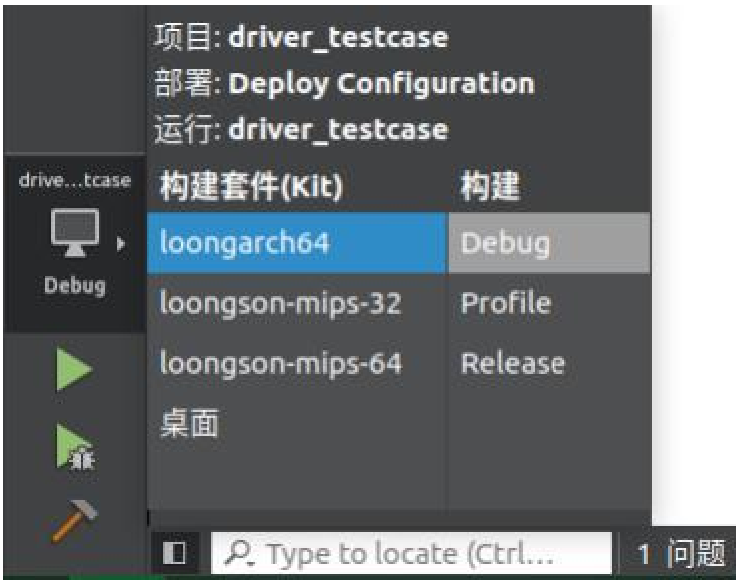
只需要到对应的阐述文件夹里面找到可执行程序文件，传输到板卡上执行即可。


夹
Desktop
test
Qt
build-untitled1-loongarch64-Debug
build-untitled1-unknown-Debug
untitled1
这是产出文件夹



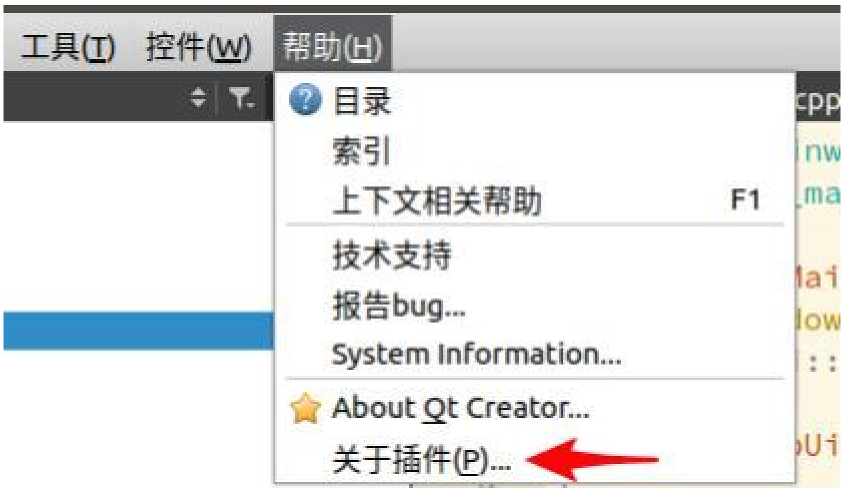
如果是移植旧项目的情况，需要选择这个编译配置的话。按照下图操作：

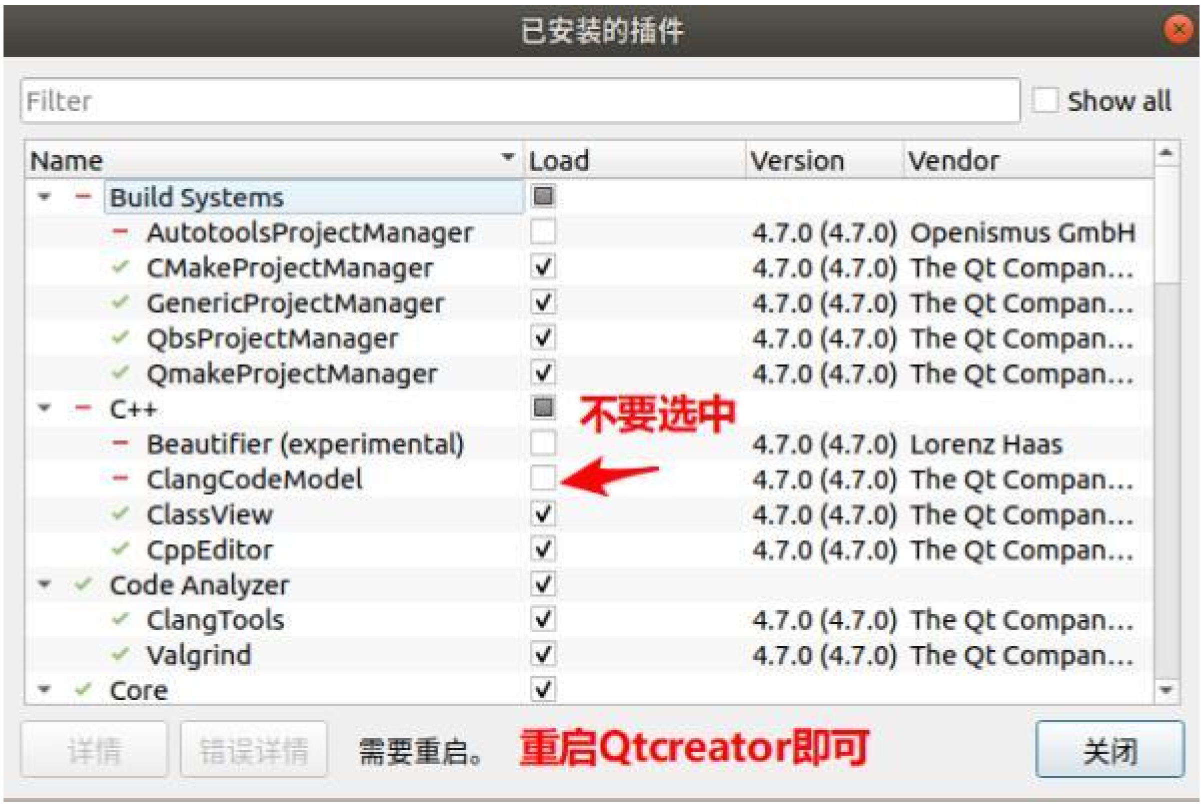




之后的操作与前文一致。

如果设置了*Qtcreator*之后，***Qtcreator*编辑的时候，代码无法高亮**，可以参考以下的解决方式





Name	Load	Version	Vendor
− Build Systems	▨		
− AutotoolsProjectManager		4.7.0 (4.7.0)	Openismus GmbH
✓ CMakeProjectManager	✓	4.7.0 (4.7.0)	The Qt Compan...
✓ GenericProjectManager	✓	4.7.0 (4.7.0)	The Qt Compan...
✓ QbsProjectManager	✓	4.7.0 (4.7.0)	The Qt Compan...
✓ QmakeProjectManager	✓	4.7.0 (4.7.0)	The Qt Compan...
− C++	▨		
− Beautifier (experimental)		4.7.0 (4.7.0)	Lorenz Haas
− ClangCodeModel		4.7.0 (4.7.0)	The Qt Compan...
✓ ClassView	✓	4.7.0 (4.7.0)	The Qt Compan...
✓ CppEditor	✓	4.7.0 (4.7.0)	The Qt Compan...
✓ Code Analyzer	✓		
✓ ClangTools	✓	4.7.0 (4.7.0)	The Qt Compan...
✓ Valgrind	✓	4.7.0 (4.7.0)	The Qt Compan...
✓ Core	✓		

8.6.3.Qt和tslib使用建议

如果使用tslib作为触控屏的使用库，比如ls2k300板中就是使用tslib库作为Qt的触摸处理库。 关于Qt和tslib的的使用，目前推荐的tslib声明为

```
export QT_QPA_FB_TSLIB=1
./driver_testcase
```

或者

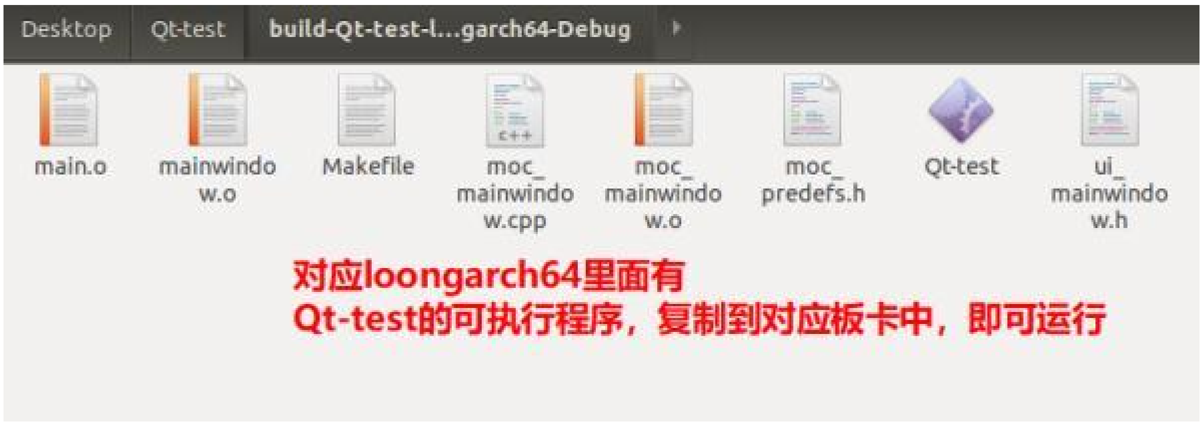
```
export QT_QPA_FB_TSLIB=1 && ./driver_testcase
```

在启动的时候，由于使用tslib作为触摸屏的Qt库，所以需要运行Qt程序之前执行`export QT_QPA_FB_TSLIB=1`的命令。如果不声明该环境变量，那么启动后，触摸可以让鼠标移动，但是会出现 不能点击按钮的bug。

并且为了更加方便，可以不用输入export QT_QPA_FB_TSLIB=1，资料包中的文件系统已经把export QT_QPA_FB_TSLIB=1写入/etc/profile里面，只需要直接运行Qt程序即可。

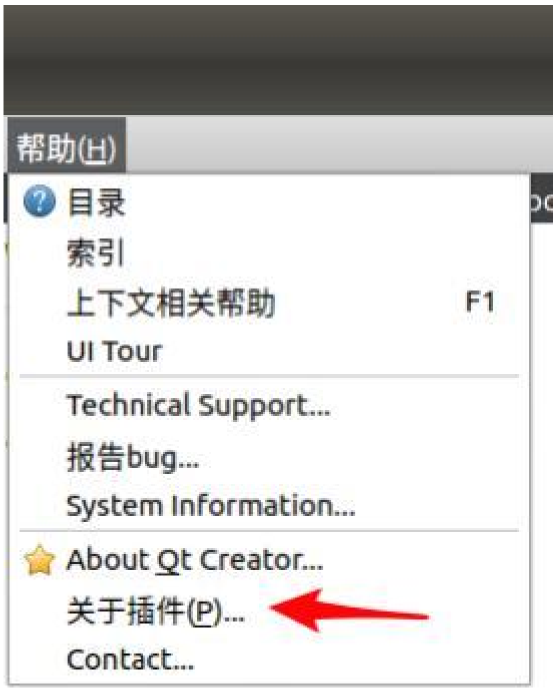
8.6.4.qtcreator代码无法高亮解决办法

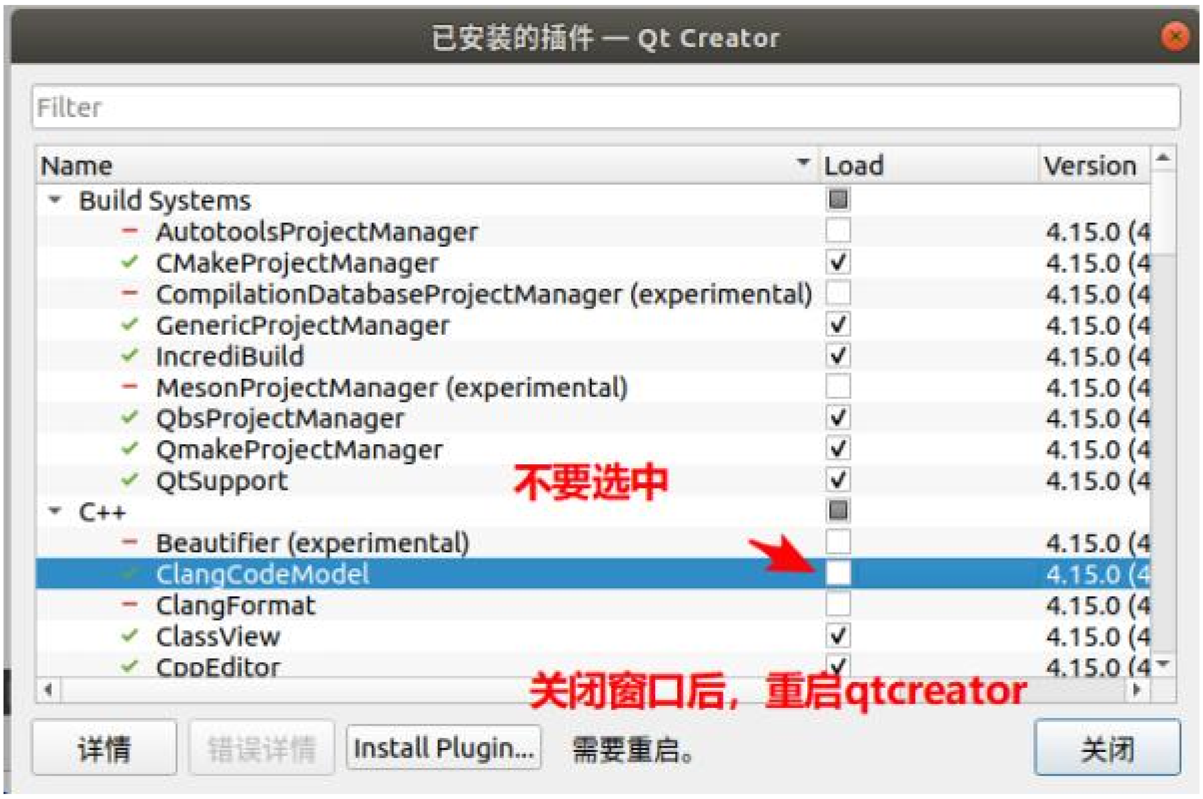
关于loongarch64的构建套件选择后， qtcreator的代码无法高亮的问题。



对应loongarch64里面有
Qt-test的可执行程序，复制到对应板卡中，即可运行

推荐按照下面的操作进行解决：

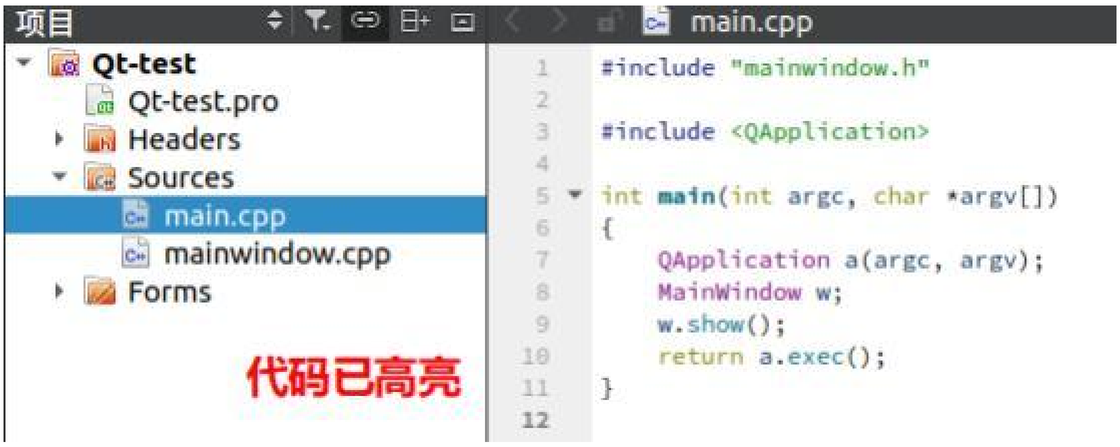



已安装的插件 — Qt Creator

Filter

Name	Load	Version
Build Systems	▨	
AutotoolsProjectManager		4.15.0 (4
CMakeProjectManager	✓	4.15.0 (4
CompilationDatabaseProjectManager (experimental)		4.15.0 (4
GenericProjectManager	✓	4.15.0 (4
IncrediBuild	✓	4.15.0 (4
MesonProjectManager (experimental)		4.15.0 (4
QbsProjectManager	✓	4.15.0 (4
QmakeProjectManager	✓	4.15.0 (4
QtSupport	✓	4.15.0 (4
C++	▨	
Beautifier (experimental)		4.15.0 (4
ClangCodeModel		4.15.0 (4
ClangFormat		4.15.0 (4
ClassView	✓	4.15.0 (4
CppEditor	✓	4.15.0 (4

问题已解决：



```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

8.7.Python库控制外设

2K300 蜂鸟板上集成了众多控制板卡外设的 python 库，使用方法如下

8.7.1.Python外设控制库

8.7.1.1.GPIO控制： RPI.GPIO

1. GPIO-75 脚输出高电平

```
import RPi.GPIO as GPIO
GPIO.setup(75, GPIO.OUT)
GPIO.output(75, 1)
```

2. GPIO-75 脚输入，并读取电平

```
import  RPi.GPIO  as  GPIO
GPIO.setup(75,    GPIO.IN)
GPIO.input(75)
```

3. GPIO-88 控制 LED 点亮、关闭

```
from RPi.GPIO import LED
led = LED(88)
led.on()
led.off()
```

4. GPIO-86 链接按键，当按键按下后做出动作

```
from RPi.GPIO import Button

def key_press():
    print("Key pressed!")

key = Button(86)
key.when_pressed = key_press
```

8.7.1.2.单线温度传感器控制： w1thermsensor

> 需要事先添加 w1-gpio 驱动，这里只演示用法

```
from w1thermsensor import W1ThermSensor
DS18B20=W1ThermSensor()
temperature = DS18B20.get_temperature()
print('%.2f'%temperature)
```

8.7.1.3. I2C-OLED SSD1306 控制： luma

将 ssd1306 OLED 接在 I2C-1 上， I2C

地址为 0x3C 显示文字：

```
from luma.core.render import canvas
from luma.oled.device import ssd1306
device = ssd1306(port=1, address=0x3C)
with canvas(device) as     draw:
    draw.text((0, 0),      'LS2K030  Pi(GD)', fill="white")
                           0
      draw.text((0, 30), 'Wellcome        to Loongson', fill="white")
```

8.7.2.Python综合DEMO

8.7.2.1. LED+BUTTON+蜂鸣器器联合控制

1. **GPIO 88、72、73** 分别接在 **红、黄、绿 LED** 上。
2. **GPIO 86、87** 接在**两个按键**上。
3. **GPIO 75** 接在 **有源蜂鸣器**上。

我们希望一个按键代表加， 一个按键代表减。初始时三个LED全灭，每加一次，就按照"绿、黄、红"的次 序多亮起一盏灯，如果此时灯全亮，则灯全灭。每减一次，就将最后一次亮起的灯灭掉，如果此时灯全 灭，则灯全点亮。除了最开始的情况，后续如果灯全灭，则蜂鸣器响BEEP。
比如加一次后亮绿灯，再加一次后绿和黄亮起，减一次后只有绿灯亮起，再减一次后灯全灭，蜂鸣器响BEEP。

```
import RPi.GPIO as GPIO
from RPi.GPIO import LED
from RPi.GPIO import Button
from time import sleep

num=0

led_g  =  LED(73)
led_y  =  LED(72)
led_r = LED(88)
leds =  [led_g, led_y, led_r]

key1 = Button(86)
key2 = Button(87)

led_r.off()
led_g.off()
led_y.off()

GPIO.setup(75, GPIO.OUT)
GPIO.output(75, 1)

def beep():
    GPIO.output(75, 0)
    sleep(0.2)
    GPIO.output(75, 1)

def light_leds():
    global num
    i=0
    for led in leds: if i
        < num:
            led.on()
        else:
            led.off() i
        += 1
    global num
    num = (num - 1) % 4
    status_check()

key1.when_pressed = key1_press
key2.when_pressed = key2_press
```

8.7.2.2.在OLED屏上显示实时温度

1. 板卡上搭载**DS18B20**单线温度传感器。
2. 将 ssd1306 OLED 接在 **I2C-1** 上, **I2C** 地址为 **0x3C**

我们希望在显示文字的同时实时显示温度

```
from w1thermsensor import W1ThermSensor
from luma.core.render import canvas
from luma.oled.device import ssd1306
from time import sleep

device = ssd1306(port=1, address=0x3C)
DS18B20=W1ThermSensor()

while True:
    temperature = DS18B20.get_temperature()

    with canvas(device) as    draw:
        draw.text((0, 0),    'LS2K0300 Pi(GD)', fill="white")
          draw.text((0, 30), 'Wellcome to Loongson', fill="white")
          draw.text((0, 45),       'Current Temp:%.2f'%temperature+' C', fill="white")

    print('%.2f'%temperature)

    sleep(0.2)
```

8.8.如何利用coredump进行调试

1. 编译时加入调试信息 （ PC 机上） 编译参数为 `-g`

```
loongarch64-linux-gnu-gcc -g 1.c
```

2. 开启core 文件 （开发板上） `ulimit -c unlimited`

查看ulimit 的所有参数设置

```
# ulimit -a
core file size          (blocks, -c) unlimited
```

```
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 3145
max locked memory       (kbytes, -l) 65536
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) unlimited
cpu time               (seconds, -t) unlimited
```

3. 查看core 文件的名字 （开发板上）

```
# cat /proc/sys/kernel/core_pattern

core
```

或者

```
# sysctl kernel.core_pattern

kernel.core_pattern = core
```

4. 示例 （开发板上）

```
# ./a.out

[  374.566878] do_page_fault(): sending SIGSEGV to a.out for invalid write access to
0000000000000000
[            374.576031]      era      =      0000000120000748      in
a.out[120000000+4000] [  374.581699] ra  = 0000000120000788
in a.out[120000000+4000] Segmentation fault (core dumped)
```

执行程序之后在当前路径生成了名为 `core` 的 coredump 文件。

```
# ls -sh

total 508K
16K a.out  488K core
```

5. 调试(PC 机上)

将板卡上运行生产的 `core` 文件复制到PC 机上，然后使用交叉工具链中的gdb 进行调试

。 调试步骤：

> 1. 启动 loongarch64-linux-gnu-gdb
> 2. 加载 二进制文件
> 3. 加载 core 文件
> 4. 查看堆栈

```
$ loongarch64-linux-gnu-gdb
GNU gdb (LoongArch GNU toolchain rc1.2 (20230615)) 8.1.50.20190122-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html> This is
free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-linux-gnu --target=loongarch64- linux
-gnu".
Type "show configuration" for configuration details. For bug
reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file a.out                ## 加载二进制文件
Reading symbols from a.out...done.
(gdb) core-file ./core          ## 加载core 文件
warning: exec file is newer than core file. [New
LWP 473]
warning: Section `.reg2/473' in core file too small.
warning: Could not load shared library symbols for /lib/loongarch64-linux-
gnu/libc.so.6.
Do you need "set solib-search-path" or "set sysroot"? Core was
generated by `./a.out'.
Program terminated with signal SIGSEGV, Segmentation fault.
warning: Section `.reg2/473' in core file too small.
#0  0x0000000120000748 in change_val (p=0x0) at 1.c:7   ## 从这里可知是给空指针赋
值后导致的
7           *p = 10;
(gdb) bt                    ## 查看堆栈
#0  0x0000000120000748 in change_val (p=0x0) at 1.c:7
#1  0x0000000120000788 in main (argc=1, argv=0x7ffbcbc0e8) at 1.c:13
(gdb)
```

示例源代码如下

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void change_val(char *p) {
        *p = 10;
```

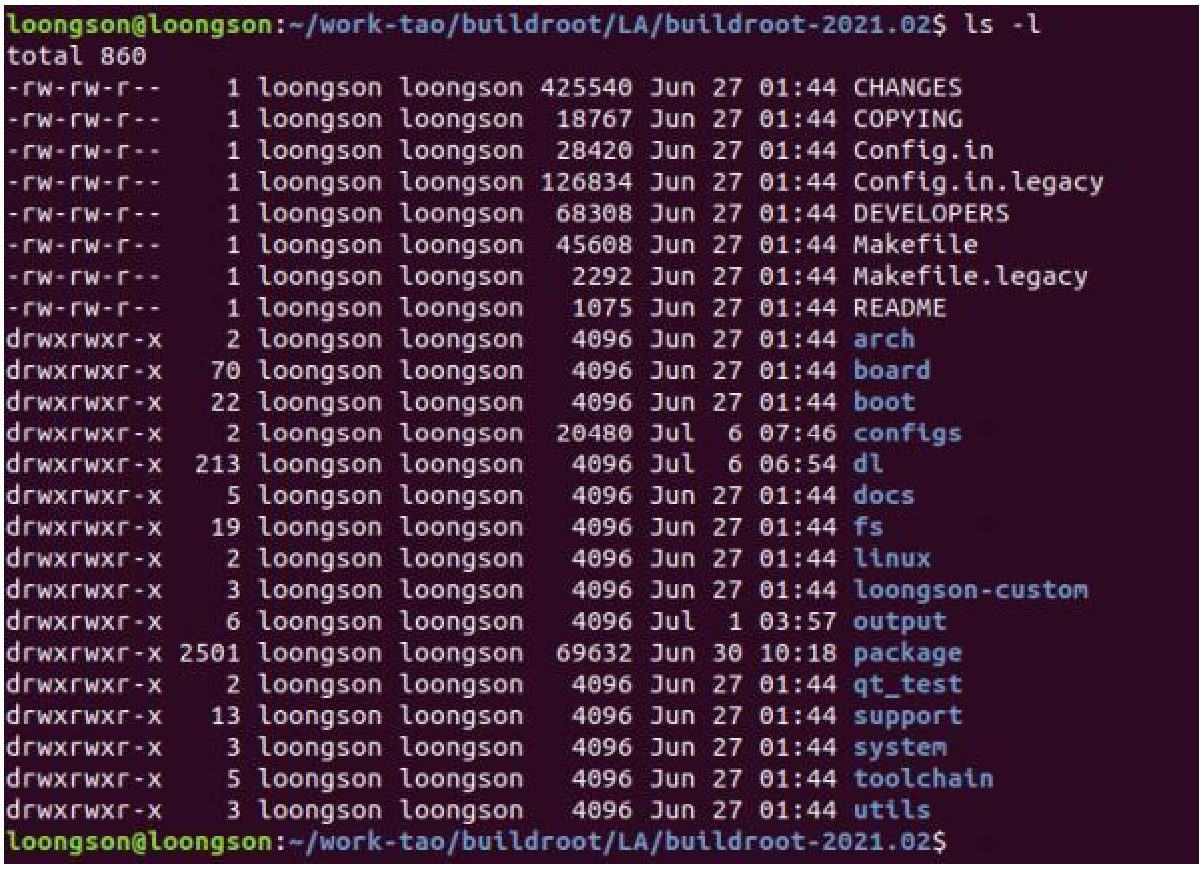
```
int main(int argc, char *argv[]) {
        char *p = NULL;
        change_val(p);
        printf("p:%d\n", *p);
        return 0;
```

九、文件系统定制

9.1.buildroot

buildroot的使用手册可以在其官网中查阅。

buildroot的源码的文件夹结构如下，注意可能与官网有所不同，因为添加了一些自定义的功能：



```
loongson@loongson:~/work-tao/buildroot/LA/buildroot-2021.02$ ls -l
total 860
-rw-rw-r--    1 loongson loongson 425540 Jun 27 01:44 CHANGES
-rw-rw-r--    1 loongson loongson  18767 Jun 27 01:44 COPYING
-rw-rw-r--    1 loongson loongson  28420 Jun 27 01:44 Config.in
-rw-rw-r--    1 loongson loongson 126834 Jun 27 01:44 Config.in.legacy
-rw-rw-r--    1 loongson loongson  68308 Jun 27 01:44 DEVELOPERS
-rw-rw-r--    1 loongson loongson  45608 Jun 27 01:44 Makefile
-rw-rw-r--    1 loongson loongson   2292 Jun 27 01:44 Makefile.legacy
-rw-rw-r--    1 loongson loongson   1075 Jun 27 01:44 README
drwxrwxr-x    2 loongson loongson   4096 Jun 27 01:44 arch
drwxrwxr-x   70 loongson loongson   4096 Jun 27 01:44 board
drwxrwxr-x   22 loongson loongson   4096 Jun 27 01:44 boot
drwxrwxr-x    2 loongson loongson  20480 Jul  6 07:46 configs
drwxrwxr-x  213 loongson loongson   4096 Jul  6 06:54 dl
drwxrwxr-x    5 loongson loongson   4096 Jun 27 01:44 docs
drwxrwxr-x   19 loongson loongson   4096 Jun 27 01:44 fs
drwxrwxr-x    2 loongson loongson   4096 Jun 27 01:44 linux
drwxrwxr-x    3 loongson loongson   4096 Jun 27 01:44 loongson-custom
drwxrwxr-x    6 loongson loongson   4096 Jul  1 03:57 output
drwxrwxr-x 2501 loongson loongson  69632 Jun 30 10:18 package
drwxrwxr-x    2 loongson loongson   4096 Jun 27 01:44 qt_test
drwxrwxr-x   13 loongson loongson   4096 Jun 27 01:44 support
drwxrwxr-x    3 loongson loongson   4096 Jun 27 01:44 system
drwxrwxr-x    5 loongson loongson   4096 Jun 27 01:44 toolchain
drwxrwxr-x    3 loongson loongson   4096 Jun 27 01:44 utils
loongson@loongson:~/work-tao/buildroot/LA/buildroot-2021.02$
```

下表将解释有关文件夹的作用：

表9－1 buildroot目录文件夹说明表

文件夹名字	作用
board	保存了和板卡相关的信息 可以前往./board/loongson里面可以看见一些针对龙芯板卡的文件，在后文详细 解释
configs	存放编译配置的文件夹
dl	编译的包的源码存放目录，没有编译之前，此文件夹中就存放了一些预下载的包，等要编译的时候可以跳过下载的阶段
output	这是输出文件夹 ./output/image文件夹里面的rootfs.tar.gz和rootfs.img文件系统
package	关于要编译的包是如何编译的(.mk文件)，并且如何在Kconfig里面定义(Config.in 文件)。 还有.hash文件，里面会记录下载的包的检验值，用于校验源码。此文件可有可 无

文件夹名字	作用
loongson-custom	目前里面包含了一个脚本文件，这个脚本文件是作用于编译LA架构的系统时，对 编译机的一次文件部署。部署的文件是config.guess和config.sub。如果是部署机是第一次编译LA架构的系统，那么需要运行此脚本（非root用户 下）

9.1.1.buildroot编译

编译之前检查请编译机的安装环境，执行命令如下：

```
sudo apt install gcc cmake tcl libtool g++ make cpio build-essential binutils libncurses5-dev hgsvn
```

如果还有其他包提示没找到，请按实际情况安装即可。

运行 `./buildenv.sh 2k300`，列出能够使用的配置

```
$ ./buildenv.sh 2k300
1) loongson2k300_defconfig
2) loongson2k300_mini_dp_defconfig
3) loongson2k300_pure_busybox_defconfig
Please enter your choice:
```

表9-2 各配置区别如下

配置名	功能
loongson2k300_defconfig	开发板全量配置，有QT、 PYTHON、 GCC 、 GDB、 MAKE、 CMKAE
loongson2k300_mini_dp_defconfig	开发板常用配置，有QT、 PYTHON，无 GCC、 GDB、 MAKE、 CMAKE
loongson2k300_pure_busybox_defconfig	开发板极简配置

选择合适的配置，比如2

```
$ ./buildenv.sh 2k300
1) loongson2k300_defconfig
2) loongson2k300_mini_dp_defconfig
3) loongson2k300_pure_busybox_defconfig
Please enter your choice:2

Your select is :loongson2k300_mini_dp_defconfig

#
# configuration written to /home/loongson/datac2/niuyize/buildroot-
```

```
2021.02/.config

#
```

开始编译：

```
make -j4
```

（4代表可并行编译的任务数量，通常等于编译机的CPU数量，按实际情况的物理CPU核数调整即可优化 编译速度）

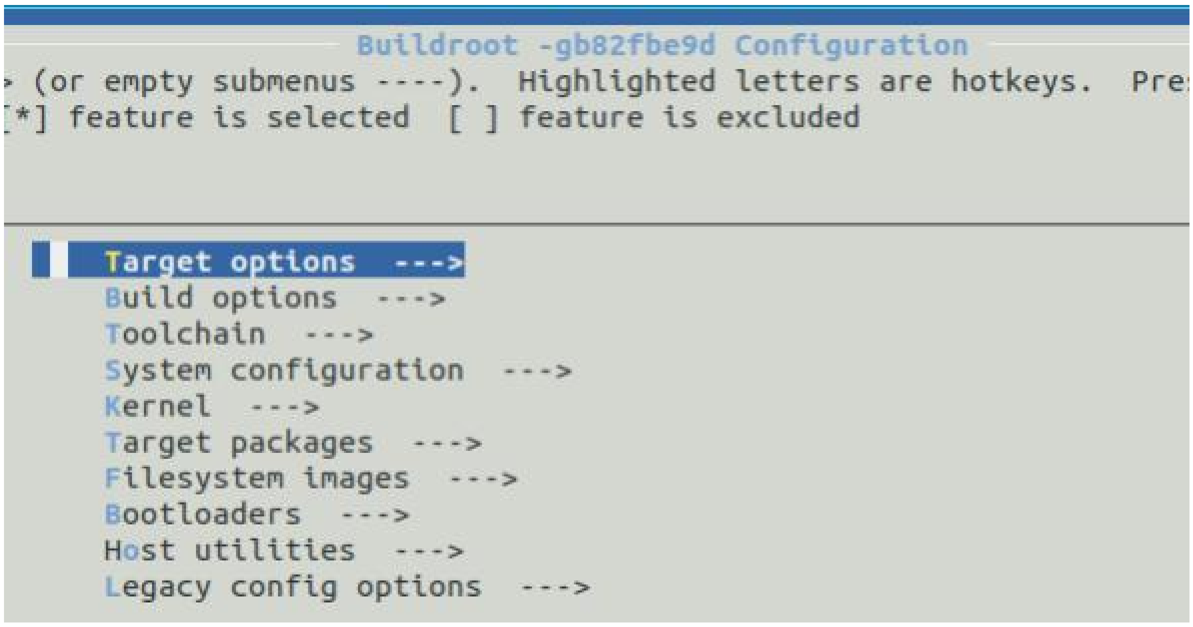
如果需要配置包那么在编译前请执行命令：

```
make menuconfig
```

相关配置项目的说明请看后文，配置完之后，执行编译命令即可等待完成编译。

9.1.2.buildroot menuconfig配置项说明

make menuconfig运 行 后 如 下 图：



```
                 Buildroot -gb82fbe9d Configuration
> (or empty submenus ----).  Highlighted letters are hotkeys.  Pre
[*] feature is selected  [ ] feature is excluded

      Target options  --->
      Build options  --->
      Toolchain  --->
      System configuration  --->
      Kernel  --->
      Target packages  --->
      Filesystem images  --->
      Bootloaders  --->
      Host utilities  --->
      Legacy config options  --->
```

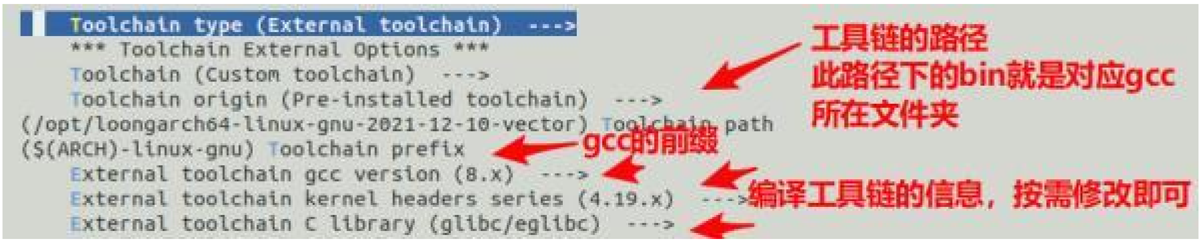
其中Target options是。 Build options是buildroot编译的文件存放，编译相关命令配置， 一般无需更 改。

表 9 - 3 make menuconfig首 层 配 置 项 说 明 表

配置项	作用
Target options	板卡上CPU的架构相关的选择， 一般无需更改
Build options	buildroot编译的文件存放，编译相关命令配置， 一般无需更改
Toolchain	交叉编译用的工具链指定，只要像前文那样配置好交叉编译工具链就行，一 般无需修改。如果修改了编译工具链，那么请进入此项，修改处可以参考下 文所述
System configuration	系统的相关配置，详细见下文所述
Kernel	无需理睬
Target packages	这是系统里面要包含的包配置，详细见下文描述

配置项	作用
Filesystem images	文件系统镜像的制作方式
Bootloaders	无需理睬
Host utilities	无需理睬
Legacy config options	无需理睬

如果需要修改编译工具链，那么请按照下图，修改里面Toolchain配置项里面的内容。下面的工具链的路 径只是展示，请按实际情况而定。

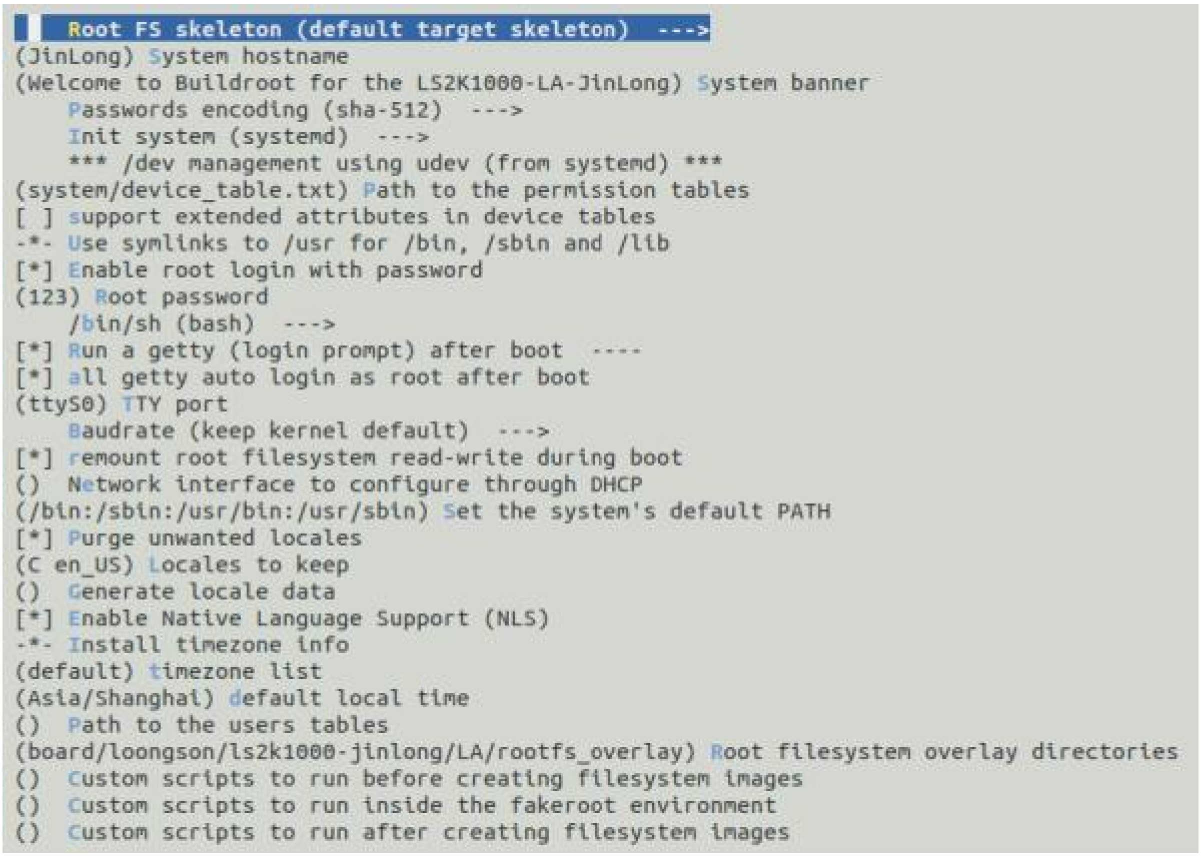


```
    Toolchain type (External toolchain)  --->
    *** Toolchain External Options ***
    Toolchain (Custom toolchain)  --->
    Toolchain origin (Pre-installed toolchain)  --->
(/opt/loongarch64-linux-gnu-2021-12-10-vector) Toolchain path
($(ARCH)-linux-gnu) Toolchain prefix
    External toolchain gcc version (8.x)  --->
    External toolchain kernel headers series (4.19.x)  --->
    External toolchain C library (glibc/eglibc)  --->
```

工具链的路径
此路径下的bin就是对应gcc所在文件夹
gcc的前缀
编译工具链的信息，按需修改即可

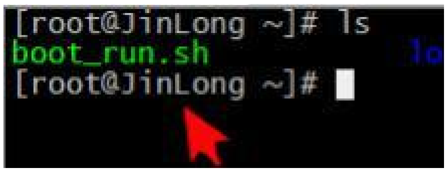
9.1.2.1.buildroot menuconfig System configuration配置项说明

关 于 System configuration配 置 项 ， 如 下 图 展 示 ：

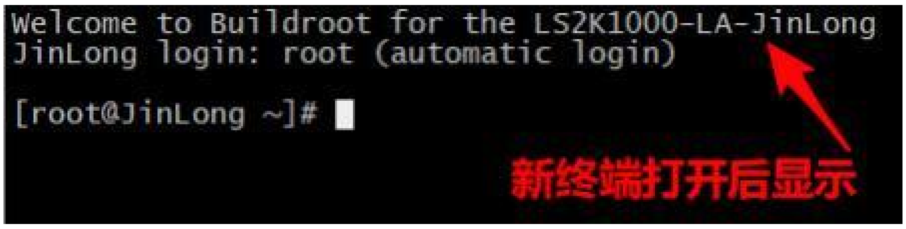


```
    Root FS skeleton (default target skeleton)  --->
(JinLong) System hostname
(Welcome to Buildroot for the LS2K1000-LA-JinLong) System banner
    Passwords encoding (sha-512)  --->
    Init system (systemd)  --->
    *** /dev management using udev (from systemd) ***
(system/device_table.txt) Path to the permission tables
[ ] support extended attributes in device tables
-*- Use symlinks to /usr for /bin, /sbin and /lib
[*] Enable root login with password
(123) Root password
    /bin/sh (bash)  --->
[*] Run a getty (login prompt) after boot  ----
[*] all getty auto login as root after boot
(ttyS0) TTY port
    Baudrate (keep kernel default)  --->
[*] remount root filesystem read-write during boot
()  Network interface to configure through DHCP
(/bin:/sbin:/usr/bin:/usr/sbin) Set the system's default PATH
[*] Purge unwanted locales
(C en_US) Locales to keep
()  Generate locale data
[*] Enable Native Language Support (NLS)
-*- Install timezone info
(default) timezone list
(Asia/Shanghai) default local time
()  Path to the users tables
(board/loongson/ls2k1000-jinlong/LA/rootfs_overlay) Root filesystem overlay directories
()  Custom scripts to run before creating filesystem images
()  Custom scripts to run inside the fakeroot environment
()  Custom scripts to run after creating filesystem images
```

System hostname的效果如下：


```
[root@JinLong ~]# ls
boot_run.sh             lo
[root@JinLong ~]# 
```

System banner的 效 果 如 下 ：



```
Welcome to Buildroot for the LS2K1000-LA-JinLong
JinLong login: root (automatic login)

[root@JinLong ~]#
```

Init System选 择 的 是 启 动 进 程， 目 前 是

systemd。 Root passwd就是root密码。

Run a getty (login prompt) after boot是屏幕也有终端启动，也就

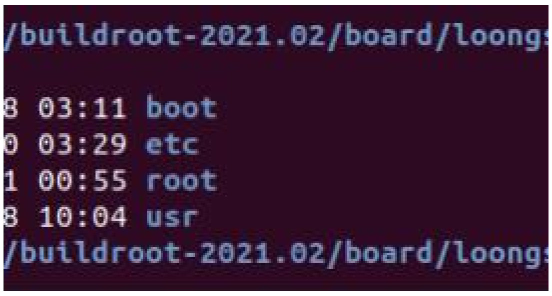
是有shell终端。 all getty auto login as root after boot终端启动不

需要登陆。

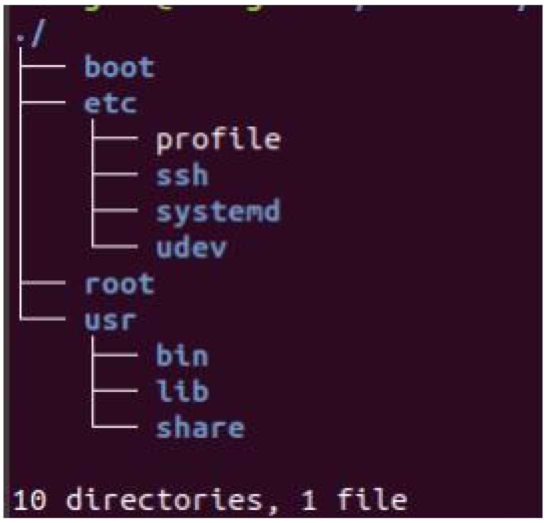
TTY port 串口调试的串口号，这个不建议修改，因为此板卡的调试串口号是固定的。

Root filesystem overlay directories 这个路径是指系统编译完成之后，最后打包之前，会把这个路径当 作一个根文件夹的映射。然后此路径下有的文件或者文件夹就会复制到编译好的系统中，然后再打包。

按照图里面的路径board/loongson/ls2k1000-jinlong/LA/rootfs_overlay，里面有四个文件夹，在打包　　之前，就会把这四个文件夹里面的文件复制到系统里面，比如/etc下面有一个profile文件，就会复制到系 统的/etc/profile下面，原本有profile文件的话，也会覆盖，按照这个为准。这样的话就能**定制一些文件**
预先打包到文件系统中。

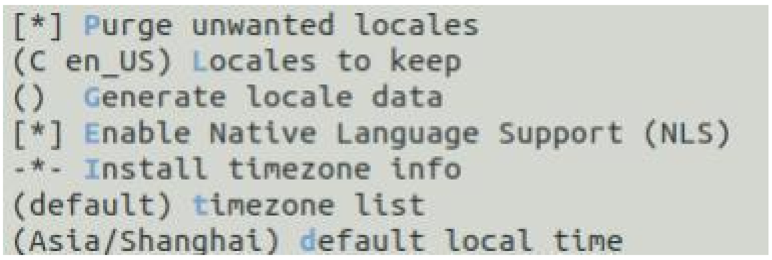


```
/buildroot-2021.02/board/loong

8 03:11 boot
0 03:29 etc
1 00:55 root
8 10:04 usr
/buildroot-2021.02/board/loong
```



```
./
├── boot
├── etc
│   ├── profile
│   ├── ssh
│   ├── systemd
│   └── udev
├── root
└── usr
    ├── bin
    ├── lib
    └── share

10 directories, 1 file
```

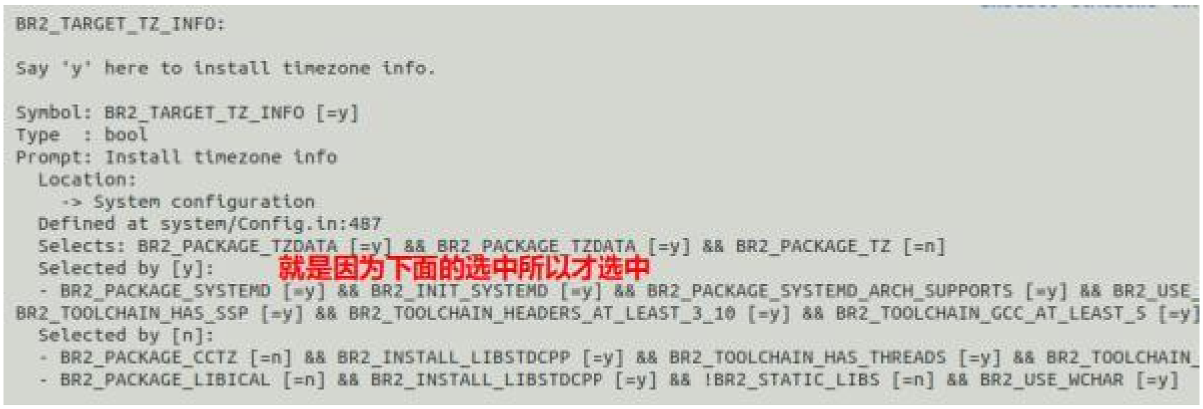
其他配置项的作用就无需理睬，也**不建议修改**。除了下图中关于时区和语言的配置。



```
[*] Purge unwanted locales
(C en_US) Locales to keep
()  Generate locale data
[*] Enable Native Language Support (NLS)
-*- Install timezone info
(default) timezone list
(Asia/Shanghai) default local time
```

值得注意的是下图的选中项是取消不了的，因为有其他配置项选中了，就必须要有这个配置项也选中。可以在该配置项中按下“?“按键(shift + /)，然后就能看见是因为那个配置项而选中这个配置项。

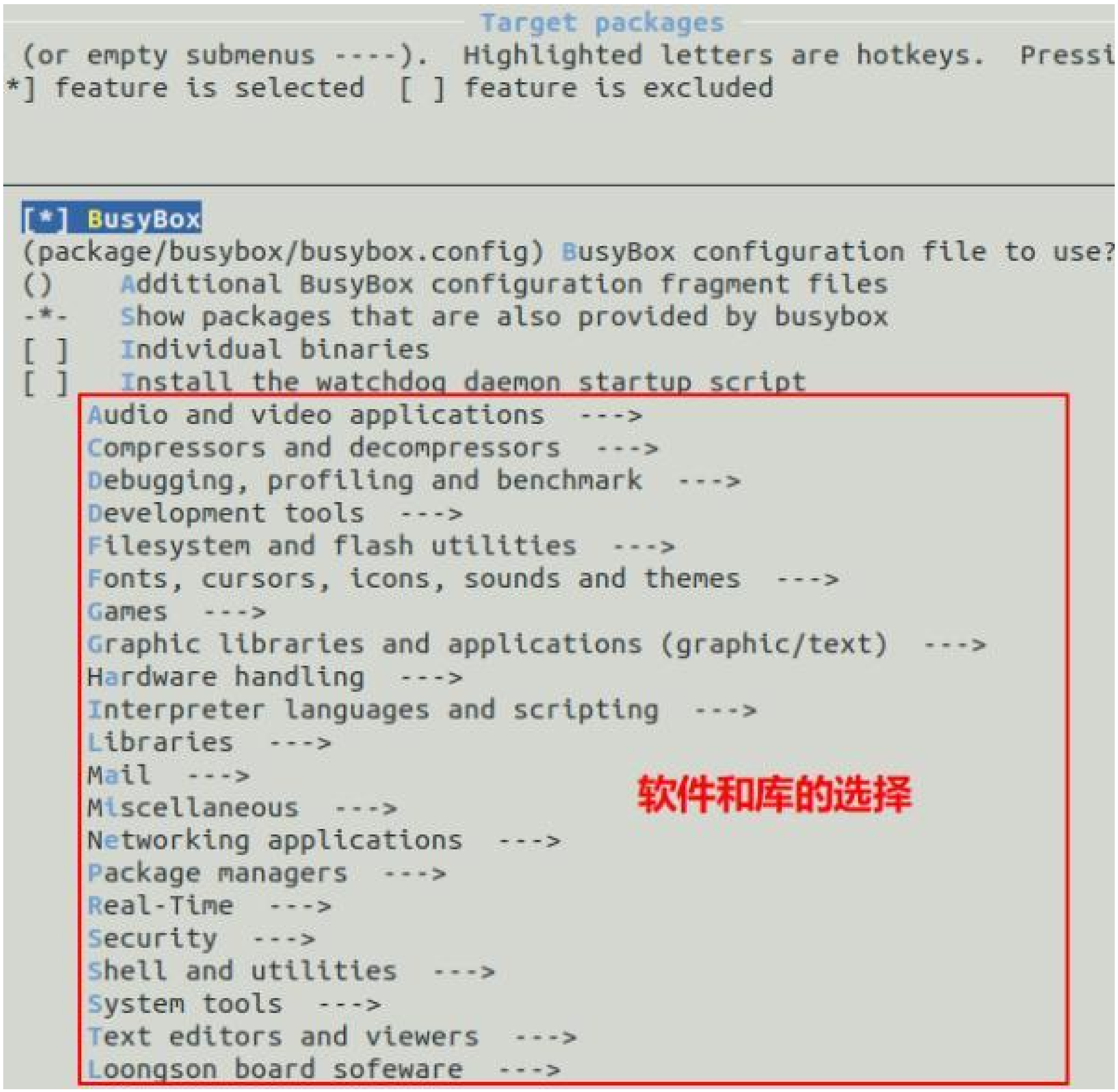
-*-



```
BR2_TARGET_TZ_INFO:

Say 'y' here to install timezone info.

Symbol: BR2_TARGET_TZ_INFO [=y]
Type  : bool
Prompt: Install timezone info
  Location:
    -> System configuration
  Defined at system/Config.in:487
  Selects: BR2_PACKAGE_TZDATA [=y] && BR2_PACKAGE_TZDATA [=y] && BR2_PACKAGE_TZ [=n]
  Selected by [y]:
  - BR2_PACKAGE_SYSTEMD [=y] && BR2_INIT_SYSTEMD [=y] && BR2_PACKAGE_SYSTEMD_ARCH_SUPPORTS [=y] && BR2_USE_
BR2_TOOLCHAIN_HAS_SSP [=y] && BR2_TOOLCHAIN_HEADERS_AT_LEAST_3_10 [=y] && BR2_TOOLCHAIN_GCC_AT_LEAST_5 [=y]
  Selected by [n]:
  - BR2_PACKAGE_CCTZ [=n] && BR2_INSTALL_LIBSTDCPP [=y] && BR2_TOOLCHAIN_HAS_THREADS [=y] && BR2_TOOLCHAIN_
  - BR2_PACKAGE_LIBICAL [=n] && BR2_INSTALL_LIBSTDCPP [=y] && !BR2_STATIC_LIBS [=n] && BR2_USE_WCHAR [=y]
```

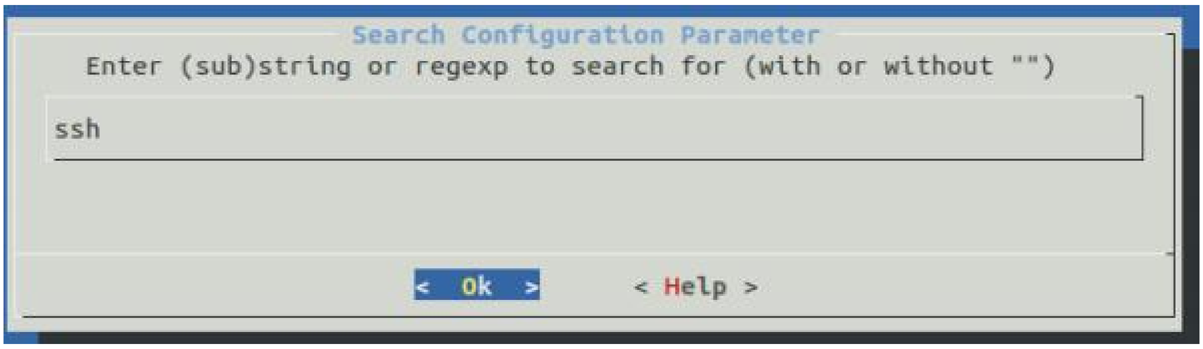
9.1.2.2.buildroot menuconfig Target packages配置项说明

Target packages配置的是要编译什么软件和库，比如说，要编译ssh到文件系统里面，那么就是在这里 面选中。下图是该配置项选中后的页面，定制软件和库的编译，需要到图中选中的框选中的那些配置项 里面找。那些配置项的名字就是大致的分类。



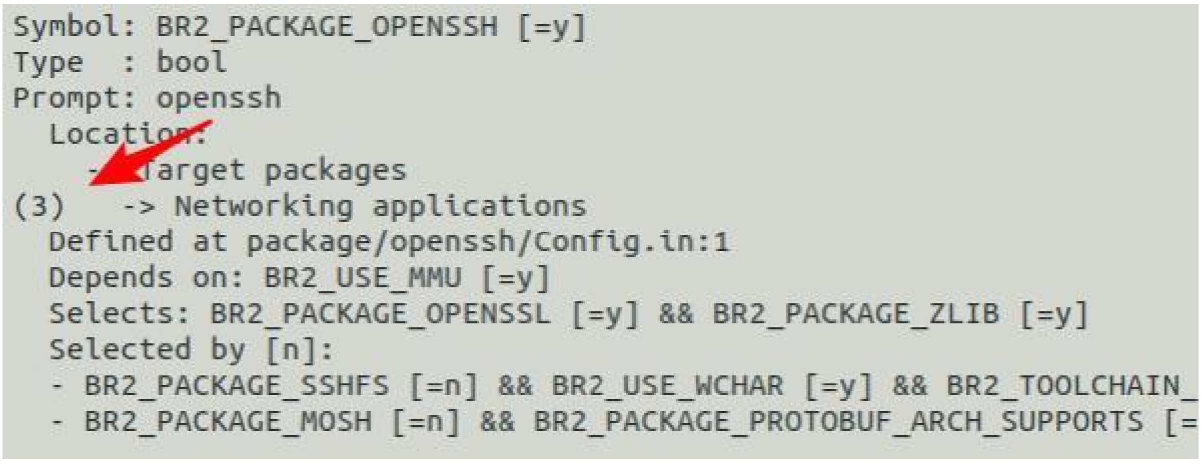
```
                         Target packages
 (or empty submenus ----).  Highlighted letters are hotkeys.  Pressi
*] feature is selected   [ ] feature is excluded


 [*] BusyBox
 (package/busybox/busybox.config) BusyBox configuration file to use?
 ()     Additional BusyBox configuration fragment files
 -*-    Show packages that are also provided by busybox
 [ ]    Individual binaries
 [ ]    Install the watchdog daemon startup script
        Audio and video applications  --->
        Compressors and decompressors  --->
        Debugging, profiling and benchmark  --->
        Development tools  --->
        Filesystem and flash utilities  --->
        Fonts, cursors, icons, sounds and themes  --->
        Games  --->
        Graphic libraries and applications (graphic/text)  --->
        Hardware handling  --->
        Interpreter languages and scripting  --->
        Libraries  --->
        Mail  --->
        Miscellaneous  --->
        Networking applications  --->
        Package managers  --->
        Real-Time  --->
        Security  --->
        Shell and utilities  --->
        System tools  --->
        Text editors and viewers  --->
        Loongson board sofeware  --->
```

也可以使用快捷查找，比如要添加ssh的编译。按下键盘的”/”键。然后输入ssh。



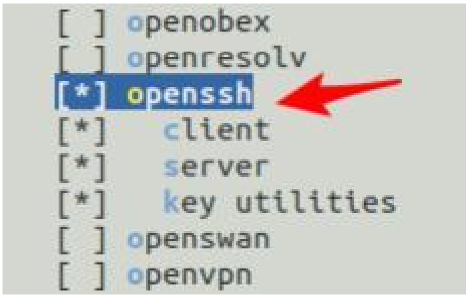
```
Search Configuration Parameter
  Enter (sub)string or regexp to search for (with or without "")

 ssh

                  < Ok >           < Help >
```

回车。然后弹出的新界面里面，可以用上下键来浏览内容，比如要找的就是openssh包，那么看到隔壁 有一个(3)，然后按下键盘上面的3，就能够跳转。



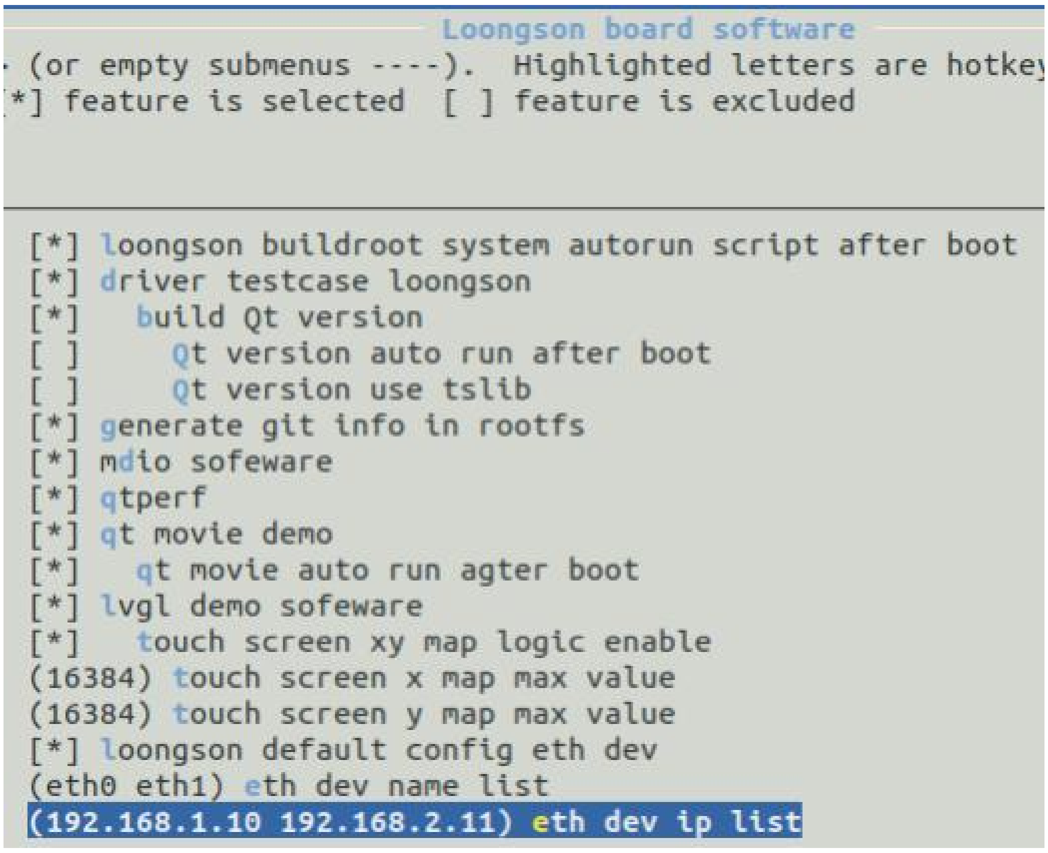
```
Symbol: BR2_PACKAGE_OPENSSH [=y]
Type  : bool
Prompt: openssh
  Location:
    -> Target packages
(3)   -> Networking applications
  Defined at package/openssh/Config.in:1
  Depends on: BR2_USE_MMU [=y]
  Selects: BR2_PACKAGE_OPENSSL [=y] && BR2_PACKAGE_ZLIB [=y]
  Selected by [n]:
  - BR2_PACKAGE_SSHFS [=n] && BR2_USE_WCHAR [=y] && BR2_TOOLCHAIN_
  - BR2_PACKAGE_MOSH [=n] && BR2_PACKAGE_PROTOBUF_ARCH_SUPPORTS [=
```

然后就会定位到对应的配置项，选中即可。



```
[ ] openobex
[ ] openresolv
[*] openssh
[*]   client
[*]   server
[*]   key utilities
[ ] openswan
[ ] openvpn
```

而在Loongson board sofeware配置项里面包含了龙芯的一些系统相关设置。



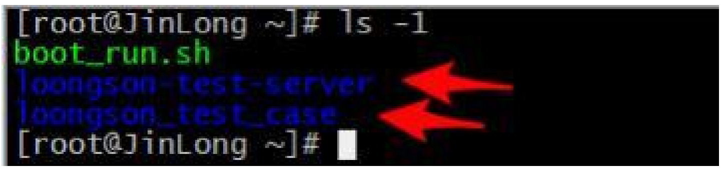
```
                    Loongson board software
- (or empty submenus ----).  Highlighted letters are hotkey
[*] feature is selected   [ ] feature is excluded


 [*] loongson buildroot system autorun script after boot
 [*] driver testcase loongson
 [*]   build Qt version
 [ ]     Qt version auto run after boot
 [ ]     Qt version use tslib
 [*] generate git info in rootfs
 [*] mdio sofeware
 [*] qtperf
 [*] qt movie demo
 [*]   qt movie auto run agter boot
 [*] lvgl demo sofeware
 [*]   touch screen xy map logic enable
 (16384) touch screen x map max value
 (16384) touch screen y map max value
 [*] loongson default config eth dev
 (eth0 eth1) eth dev name list
 (192.168.1.10 192.168.2.11) eth dev ip list
```

首先是*loongson buildroot system autorun script after boot*选项，此项对应的是前文中描述的系 统里面的

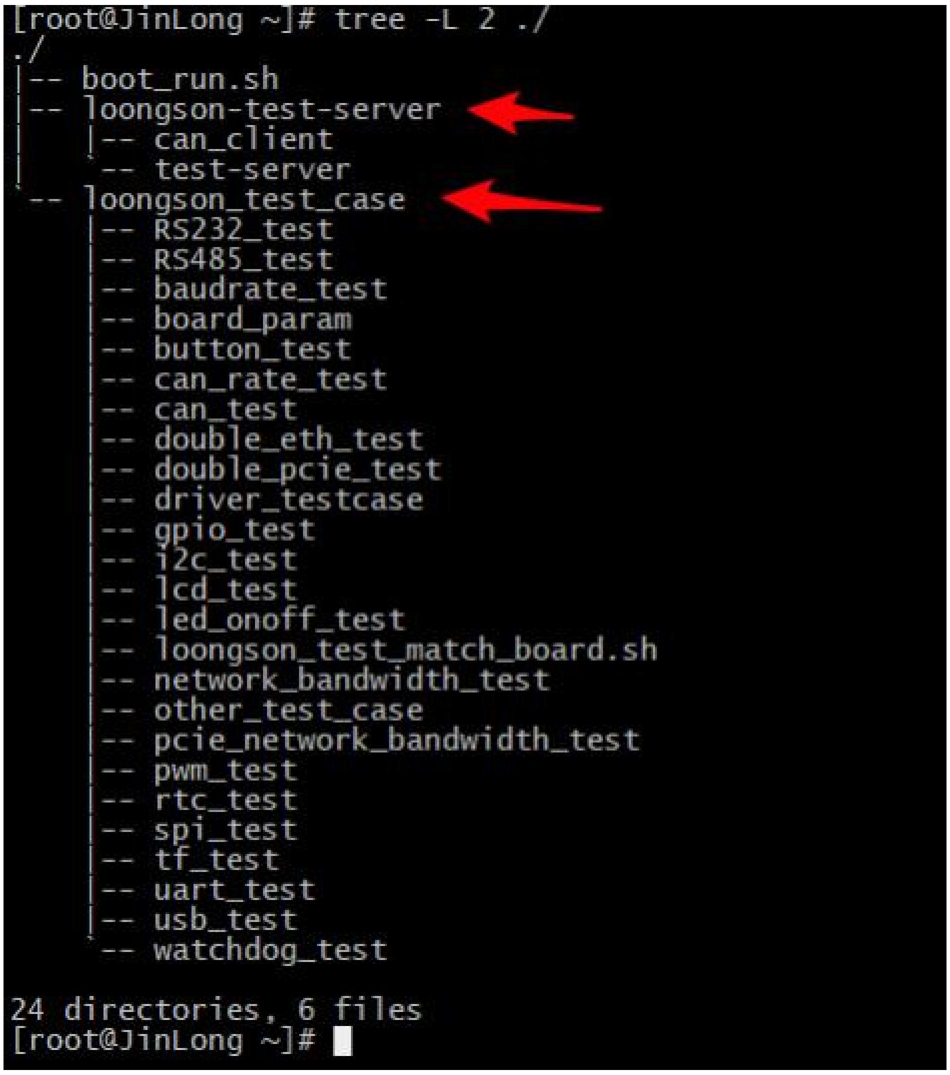
boot_run.sh的启动脚本服务的配置。

不选中则可以让此服务不存在于系统中。

此后是*driver testcase loongson*这一项，这项会制作龙芯板卡的测试用例，然后部属于*/root*下。见下图:



```
[root@JinLong ~]# ls -1
boot_run.sh
loongson-test-server
loongson_test_case
[root@JinLong ~]#
```



```
[root@JinLong ~]# tree -L 2 ./
./
|-- boot_run.sh
|-- loongson-test-server
|   |-- can_client
|   `-- test-server
`-- loongson_test_case
    |-- RS232_test
    |-- RS485_test
    |-- baudrate_test
    |-- board_param
    |-- button_test
    |-- can_rate_test
    |-- can_test
    |-- double_eth_test
    |-- double_pcie_test
    |-- driver_testcase
    |-- gpio_test
    |-- i2c_test
    |-- lcd_test
    |-- led_onoff_test
    |-- loongson_test_match_board.sh
    |-- network_bandwidth_test
    |-- other_test_case
    |-- pcie_network_bandwidth_test
    |-- pwm_test
    |-- rtc_test
    |-- spi_test
    |-- tf_test
    |-- uart_test
    |-- usb_test
    `-- watchdog_test

24 directories, 6 files
[root@JinLong ~]#
```

前往/root/loongson_test_case可以进行测试。

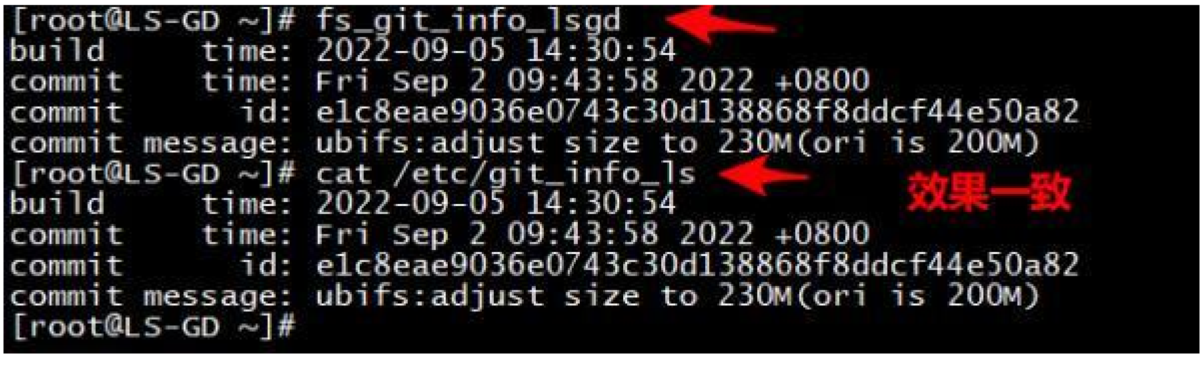
而**build Qt version**和**Qt version auto run after boot**，这两项则代表编译Qt版本的测试软件和开机 自启动Qt版本的测试软件。

如果不选中**loongson buildroot system autorun script after boot**，但是选中Qt version auto run after boot，则会创建一个driver_case.serivce的服务来达成目的。否则在boot_run.sh里面加入启动Qt 版本的测试软件的命令。

Qt测试软件所在的路径为：/root/loongson_test_case/driver_testcase

Qt version use tslib一项，则是声明此程序采用tslib来作为Qt的触摸屏输入接口，针对于ls2k500迷你 开发版，ls2k1000星云板的触摸屏，而ls2k1000星云板等采用的是USB HMI的接口，无需tslib校准。

generate git info in rootfs一项选中，则会在buildroot编译构建的时候，在生成的文件系统里面输出 一份关于编译时的git信息。使用fs_git_info_lsgd命令可以看见保存的信息，信息保存在/etc/git_info_ls文 件中，见下图。



```
[root@LS-GD ~]# fs_git_info_lsgd
build      time: 2022-09-05 14:30:54
commit     time: Fri Sep  2 09:43:58 2022 +0800
commit       id: e1c8eae9036e0743c30d138868f8ddcf44e50a82
commit message: ubifs:adjust size to 230M(ori is 200M)
[root@LS-GD ~]# cat /etc/git_info_ls
build      time: 2022-09-05 14:30:54
commit     time: Fri Sep  2 09:43:58 2022 +0800
commit       id: e1c8eae9036e0743c30d138868f8ddcf44e50a82
commit message: ubifs:adjust size to 230M(ori is 200M)
[root@LS-GD ~]#
```

mdio sofeware一项则会生成一个mdio软件，此软件用于控制网络phy芯片的寄存器，软件的用法请自 行搜索。

qtperf一项则会生成qtperf测试软件。

qt movie demo则是前面所述的logo_player，qt movie auto run agter boot一项选中后，开机自启 动logo_player。注意如果和Qt version auto run after boot一起选中，则会先启动logo_player之后再 启动driver_testcase。**但是建议不要同时选中。**

lvgl demo sofeware则会编译生成lvgl 8.2的demo，在系统中输入lvgl-demo命令即可启动

touch screen xy map logic enable、touch screen x map max value和touch screen y map max value则代表触控屏的xy是否需要映射，映射的最大值是什么。视具体情况，input_event上报的数据而 定。可以在/etc/lvgl_config文件中修改。

loongson default config eth dev代表设置默认的网口ip。eth dev name list和eth devip list是要 设置的默认的网卡设备和ip。如果使用systemd的话，那么就是按照NetworkManger的规则设置，如果 是busybox启动，那么就是设置/etc/network/interfaces文件。按照图中的格式设置，不同的网卡和ip之间只用一个空格间隔即可。

9.1.3.buildroot 编译及其产出

配置好buildroot之后，输入命令：

```
make -j4
```

（4代表可并行编译的任务数量，通常等于编译机的CPU数量，按实际情况的物理CPU核数调整即可优化 编译速度）

如果是第一次编译，那么所需时间将会很长。如果是个人PC机的话，完整编译可能需要3小时以上，并且
编译后的buildroot文件夹的大小将会达到12G左右。所以**推荐在服务器上编译**。

编译结束后，前往./output/image文件夹中即可看见编译出来的文件系统镜像。 rootfs.tar.gz和
rootfs.img是可以部署在eMMC上的文件系统部署包。具体如何部署，请看4.4EMMC使用方法中的 eMMC安装系统。

9.1.4.buildroot添加自定义包

关于builroot如何添加自定义的包， buildroot的官网有详细介绍(从第16章开始)：
https://buildroot.org/ downloads/manual/manual.html

本节将会简述如何添加一个包，也可以直接参考buildroot源码的

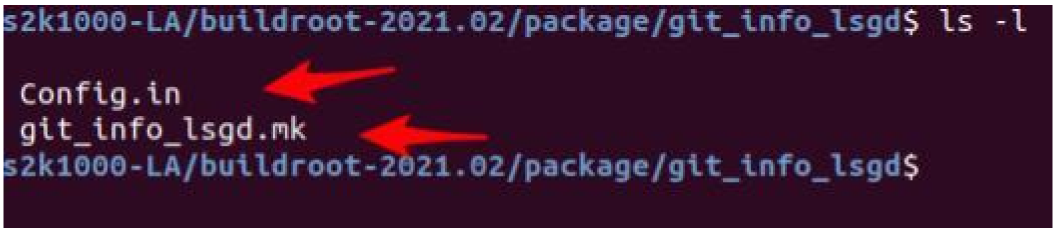
- ./package/git_info_lsgd （无编译系统，并且不是编译程序，只是生成脚本和文件）
- ./package/driver_testcase （存在cmake编译，make，Qt编译）
- ./package/mdio （无编译系统）
  对于autotools-package编译(带configure文件的源码包)，可以参考上述官网链接或者网络搜索。

对于buildroot来说，有其一套添加规则。以git_info_lsgd为例：
首先肯定需要在package文件夹里面新建一个文件夹，建议包含字母、数字、下划线。 也就是./package/git_info_lsgd文件夹的由来。

在这个文件夹里面最重要的就是两个文件， 一个Config.in文件，一个.mk文件。 文件的命名规则

表9-4 ./package/*构建包的文件夹文件命名规则表

文件	规则
Config.in	只能是Config.in
.mk文件	和文件夹同名，后缀是.mk



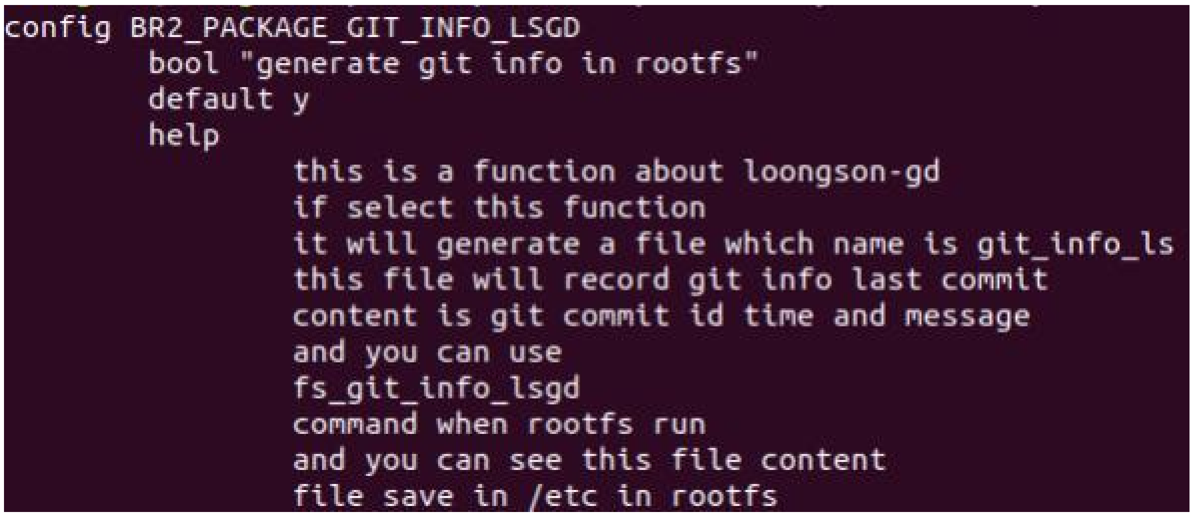
```
s2k1000-LA/buildroot-2021.02/package/git_info_lsgd$ ls -l

 Config.in
 git_info_lsgd.mk
s2k1000-LA/buildroot-2021.02/package/git_info_lsgd$
```

9.1.4.1.Config.in文件解析

Config.in文件提供的是构建选项，直观点说就是make menuconfig的可选项。然后Config.in里面提供的 选项，选中了，在.mk文件里面将会作为构建参数。这个参数的意思不仅仅可以作用于编译源码，还可以 包括，要不要执行某些动作，编译源码的某些参数。

buildroot里面的构建过程，不仅仅是编译源码，还大致包括源码包的获取，源码包的解压，编译之后产 出文件的安装。而上述Config.in里面提供的选项，则可以作用于这个过程里面的每个小过程。而怎么发 挥作用，则需要在.mk文件中规定。

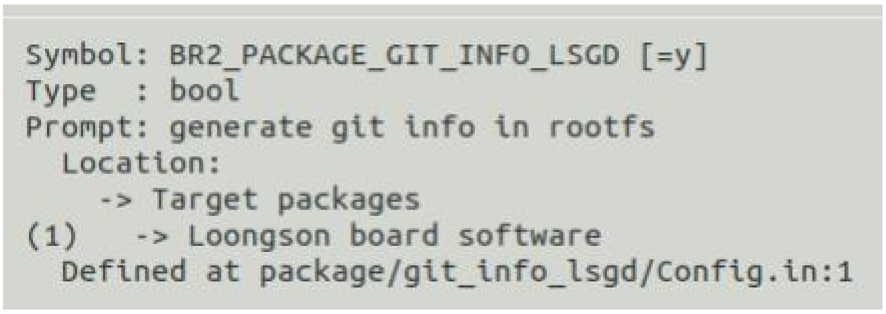
比如git_info_lsgd的Config.in文件



```
config BR2_PACKAGE_GIT_INFO_LSGD
        bool "generate git info in rootfs"
        default y
        help
                this is a function about loongson-gd
                if select this function
                it will generate a file which name is git_info_ls
                this file will record git info last commit
                content is git commit id time and message
                and you can use
                fs_git_info_lsgd
                command when rootfs run
                and you can see this file content
                file save in /etc in rootfs
```

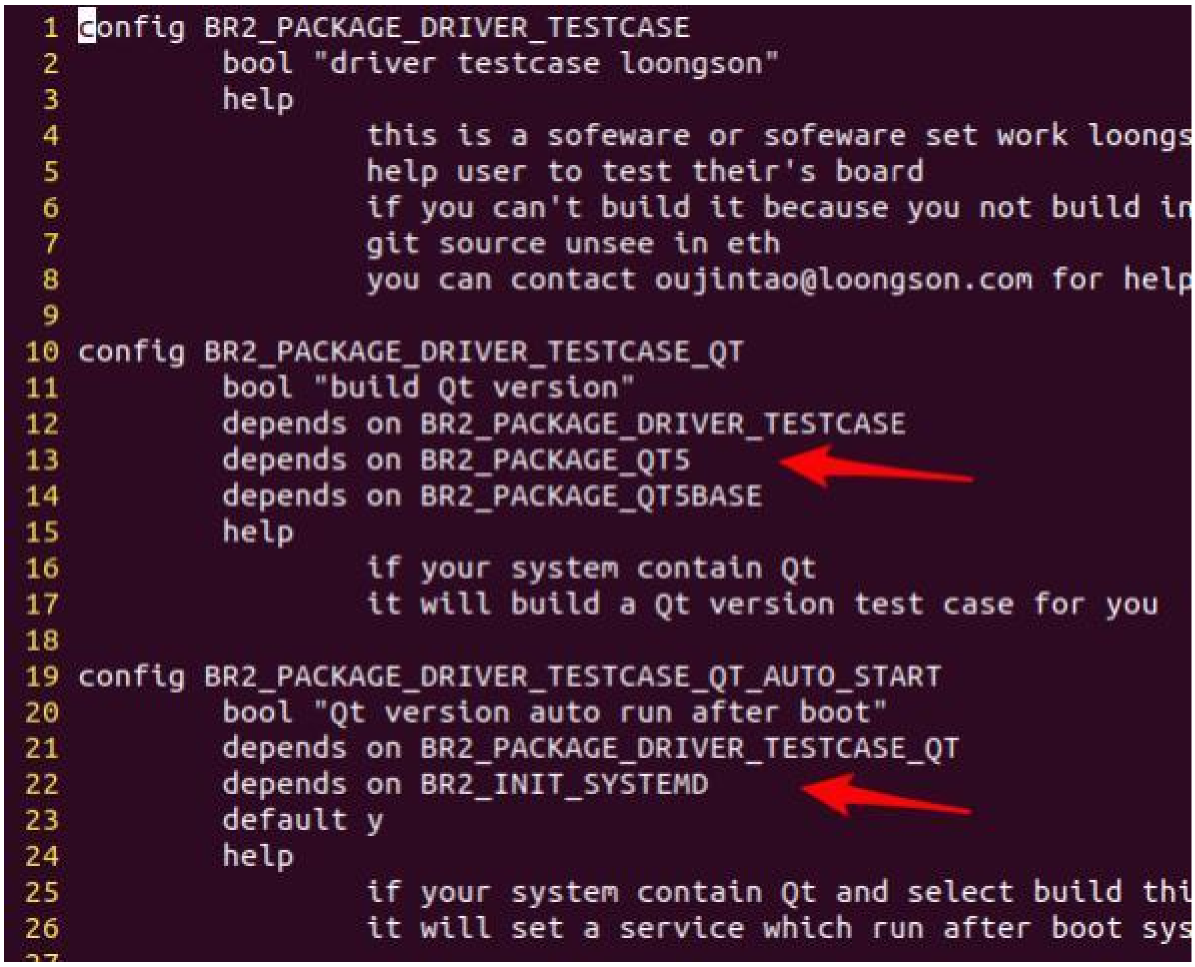
BR2_PACKAGE_GIT_INFO_LSGD这是这个选项的名字。最终效果在make menuconfig里面就能找到这 个选项，然后可以去选中。 default y则是默认选中， **help及其后面的文字**就是这个选项的提示语，包含 了这个选项的意义。 bool代表这个选项的值是bool值，还有其他类型的值，可以查看buildroot的官方手 册，后面的字符串则是选项的显示文本。

BR2_PACKAGE_GIT_INFO_LSGD 中 的 BR2_PACKAGE 建 议 是 固 定 的 ， 而 GIT_INFO_LSGD则是包名（小 写字母变大写字母），也就是说**BR2_PACKAGE_包名**这个选项一定要有，这是决定要不要构建这个包的 可选项。



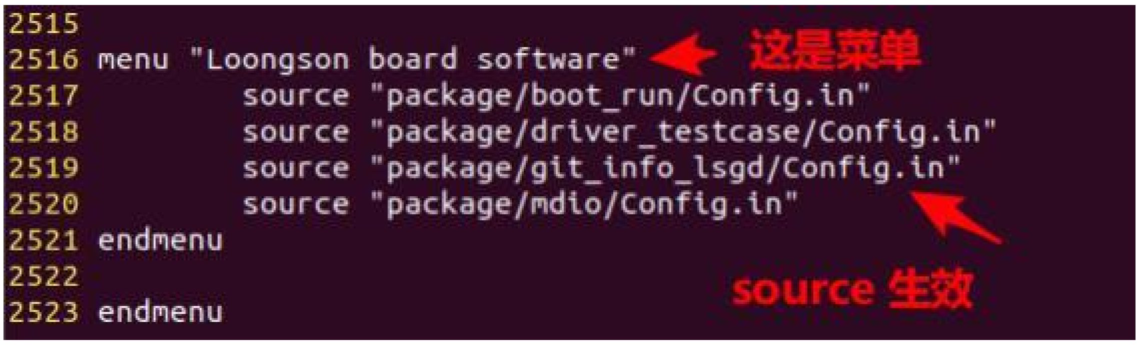
```
Symbol: BR2_PACKAGE_GIT_INFO_LSGD [=y]
Type  : bool
Prompt: generate git info in rootfs
  Location:
    -> Target packages
(1)   -> Loongson board software
  Defined at package/git_info_lsgd/Config.in:1
```

还有一个字段为*depends on*。这个是这个选项的前置条件，以 *./package/driver_testcase/Config.in*为 例，如果*depends on*的条件不成立，那么对应的选项则不会出现在*make menuconfig*中并且不选中。以 *BR2_PACKAGE_DRIVER_TESTCASE_QT*来说，如果系统中不构建*Qt5*，那么这个选项就没有要出现的意 义，选中为*n*即可。

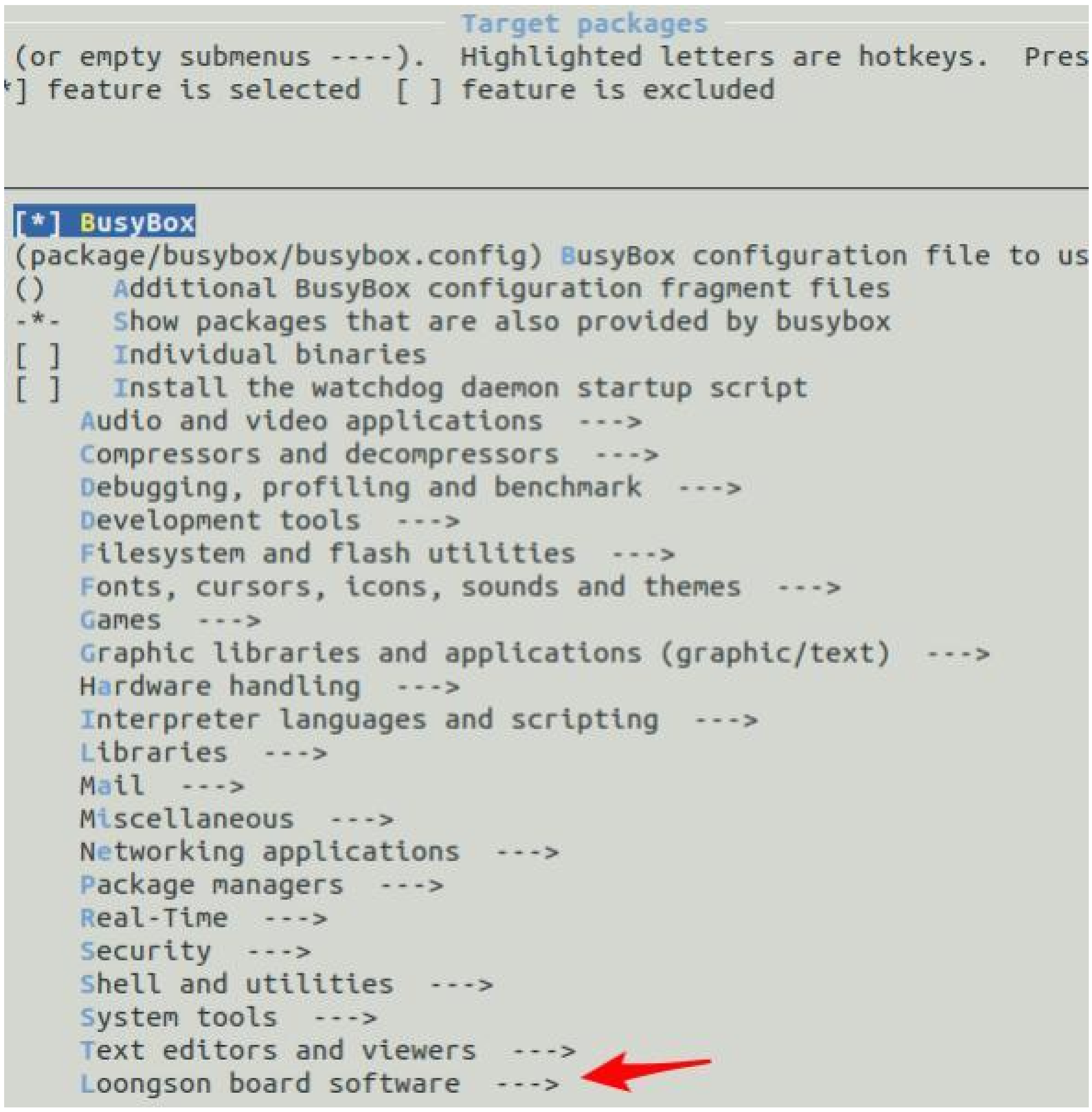
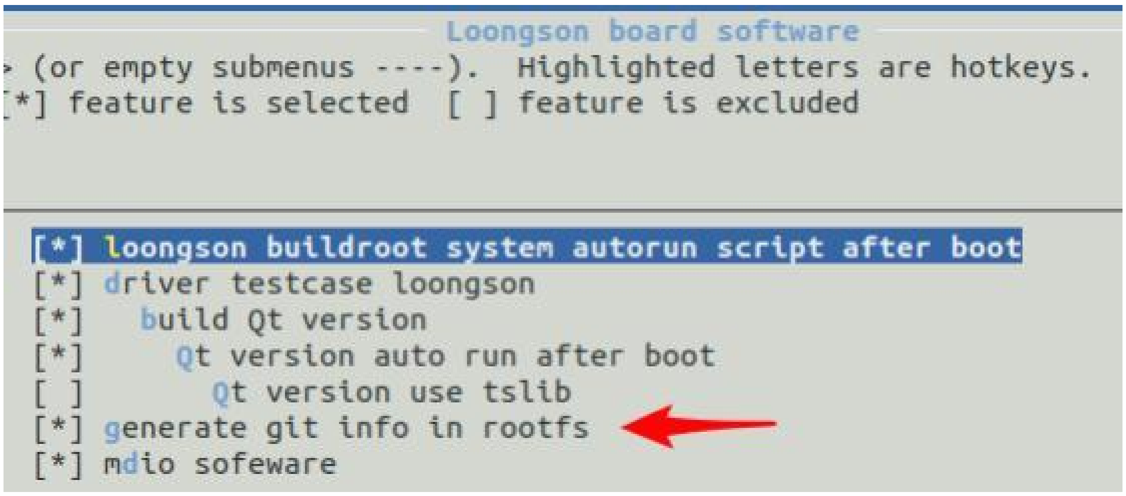


```
config BR2_PACKAGE_DRIVER_TESTCASE
        bool "driver testcase loongson"
        help
                this is a sofeware or sofeware set work loongs
                help user to test their's board
                if you can't build it because you not build in
                git source unsee in eth
                you can contact oujintao@loongson.com for help

config BR2_PACKAGE_DRIVER_TESTCASE_QT
        bool "build Qt version"
        depends on BR2_PACKAGE_DRIVER_TESTCASE
        depends on BR2_PACKAGE_QT5
        depends on BR2_PACKAGE_QT5BASE
        help
                if your system contain Qt
                it will build a Qt version test case for you

config BR2_PACKAGE_DRIVER_TESTCASE_QT_AUTO_START
        bool "Qt version auto run after boot"
        depends on BR2_PACKAGE_DRIVER_TESTCASE_QT
        depends on BR2_INIT_SYSTEMD
        default y
        help
                if your system contain Qt and select build thi
                it will set a service which run after boot sys
```

如果只是添加了*Config.in*文件， *make menuconfig*里面其实是找不到的， 需要在 *./package/Config.in* 添加相应的声明， 见下图， *menu*那个是菜单声明， *Loongson board sofeware*是菜单名， 随后*source* 的*Config.in*文件， 将会把对应的选项加上。然后*make menuconfig*里面才会有对应的选项。



```
2515
2516 menu "Loongson board software"
2517         source "package/boot_run/Config.in"
2518         source "package/driver_testcase/Config.in"
2519         source "package/git_info_lsgd/Config.in"
2520         source "package/mdio/Config.in"
2521 endmenu
2522
2523 endmenu
```


```
                               Target packages
(or empty submenus ----).  Highlighted letters are hotkeys.  Pres
*] feature is selected  [ ] feature is excluded


[*] BusyBox
(package/busybox/busybox.config) BusyBox configuration file to us
()    Additional BusyBox configuration fragment files
-*-   Show packages that are also provided by busybox
[ ]   Individual binaries
[ ]   Install the watchdog daemon startup script
    Audio and video applications  --->
    Compressors and decompressors  --->
    Debugging, profiling and benchmark  --->
    Development tools  --->
    Filesystem and flash utilities  --->
    Fonts, cursors, icons, sounds and themes  --->
    Games  --->
    Graphic libraries and applications (graphic/text)  --->
    Hardware handling  --->
    Interpreter languages and scripting  --->
    Libraries  --->
    Mail  --->
    Miscellaneous  --->
    Networking applications  --->
    Package managers  --->
    Real-Time  --->
    Security  --->
    Shell and utilities  --->
    System tools  --->
    Text editors and viewers  --->
    Loongson board software  --->
```

```
                      Loongson board software
> (or empty submenus ----).  Highlighted letters are hotkeys.
[*] feature is selected  [ ] feature is excluded


 [*] loongson buildroot system autorun script after boot
 [*] driver testcase loongson
 [*]   build Qt version
 [*]     Qt version auto run after boot
 [ ]       Qt version use tslib
 [*] generate git info in rootfs
 [*] mdio sofeware
```

9.1.4.2.mk文件解析

前文阐述了Config.in的作用是提供可选项，而.mk文件是根据可选项，来决定是否构建，怎么构建。其中 怎么构建可能就需要可选项的结果来决定。

BR2_PACKAGE_包名选中时， .mk文件里面的内容才会作用于buildroot的构建过程，也就是make的时 候。

表9-5 .mk文件部分符号解释表

符号	含义
包名SITE 包名 SITE_METHOD 等	指定源码包的下载方式 见官方手册的18.6.2 节
包名_EXTRACT_CMDS	说明怎么解压源码包
包名_BUILD_CMDS	说明怎么构建源码包
包名 _INSTALL_TARGET_C MDS	说明怎么部署输出文件
target TARGET	代表输出的文件系统
host	交叉编译中 host机器的相 关 比如./output/host 文件夹
$(@D)	构建时，会在./output/build里面生成一个包名的文件夹， $(@D) 是这个文件夹的路径
$(TARGET)	通常指代./output/target 的绝对路径
$(TOPDIR)	通常指代buildroot的绝对路径

.mk的内容也有一定的规律。见下图和分析。

以git_info_lsgd为例：例子中变量和define的命令，都是以**包名的大写形式**+特定的一些后缀，这些特定 的定义是一定要实现的。也可以添加其他的定义，但是buildroot不会自动解析。

第1-5行是注释，无需理睬，但是其他文件的也是这样写，所以也这样写

第7-8行是声明了源码包从哪里来。本例中，源码包从本地获取，也就是buildroot的./dl/git_info_lsgd/git_info_lsgd.tar.gz。 SITE_METHOD则是声明获取方式， file则是本地的一个文件。 其他的方式可以参考driver_testcase或者官方手册的18.6.2节

```
GIT_INFO_LSGD_SITE=$(TOPDIR)/dl/git_info_lsgd/git_info_lsgd.tar.gz

GIT_INFO_LSGD_SITE_METHOD=file
```

第10-12行则是说明了如何解压源码包， $(@D)通常就是./output/build/包名文件夹，按照本例，则 为./output/build/git_info_lsgd。

```
define GIT_INFO_LSGD_EXTRACT_CMDS

      tar -zxf $(GIT_INFO_LSGD_SITE) -C $(@D)

endef
```

第14-17行则说明如何构建输出文件，其中不仅仅是编译代码，也可以只是像本例，只是执行一个脚本 （脚本的逻辑是收集git信息），然后给另外一个脚本（fs_git_info_lsgd）添加可执行权限。

```
define GIT_INFO_LSGD_BUILD_CMDS
	cd $(@D) && chmod a+x git-info-get.sh &&  ./git-info-get.sh cd
	$(@D) && chmod a+x fs_git_info_lsgd
endef
```

第19-22行说明了如何安装输出文件在文件系统中， *git_info_ls*文件是*git-info-get .sh*的输出文件，
*fs_git_info_lsgd*在前面*GIT_INFO_LSGD_BUILD_CMDS*执行时已经添加了可执行权限，放在文件系统 的./*usr/bin*里面就能够在文件系统运行时直接输入***fs_git_info_lsgd***运行，看到*git*信息。

$(TARGET_DIR)一般指的是./*output/target*文件夹。

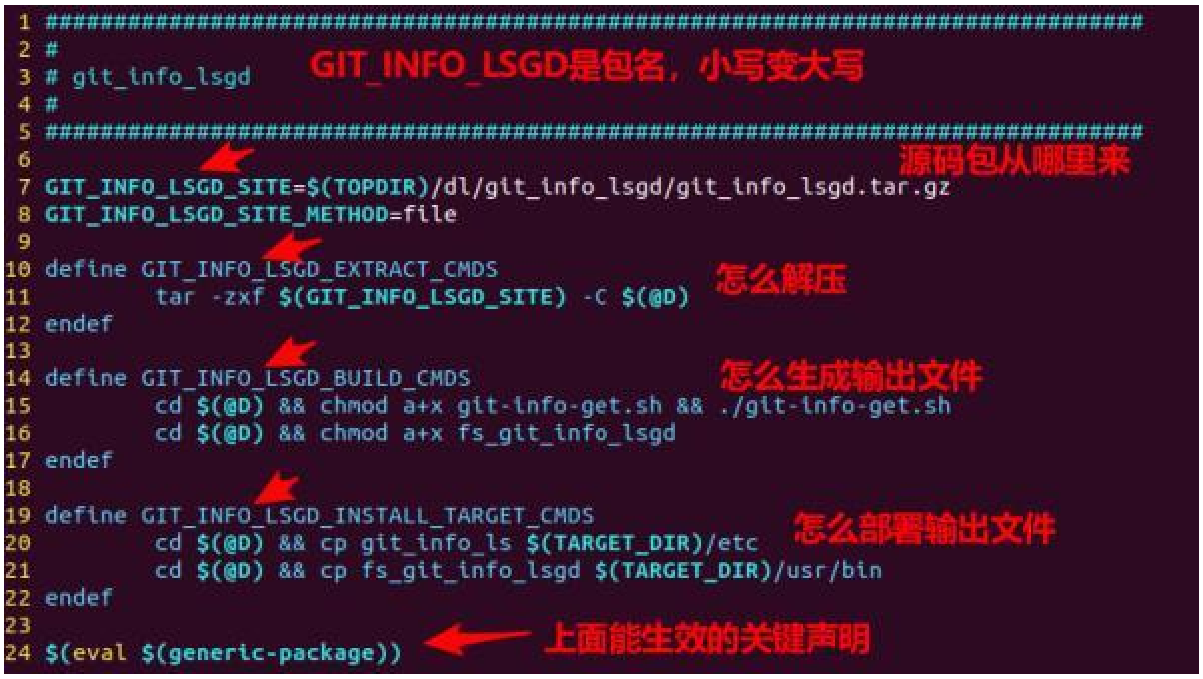
```
define GIT_INFO_LSGD_INSTALL_TARGET_CMDS
      cd $(@D) && cp git_info_ls $(TARGET_DIR)/etc
      cd $(@D) && cp fs_git_info_lsgd $(TARGET_DIR)/usr/bin endef
```

最后一行十分重要，这个让前面的设置生效的根本。这里指定了构建这个包的通用流程。

*buildroot*是自动化构建包。而手动构建一个包需要先下载源码，解压源码，编译源码，安装输出文件。 而下载源码的方式已经在第7-8行那里声明了。而编译源码的方式有很多种，采用*makefile*、采用
Cmake、才有*autotools*等方式，还有人为指定的编译方式。而这一行则是指定为人为指定的编译方式。

前面的*BUILD_CMDS*和*INSTALL_TARGET_CMDS*这两个定义的名字不是随便定的，这是*buildroot*会自动 寻找，然后执行。如果这里为**$(eval $(autotools-package))**，则将其认为是./*configure makemake* *install*的那种方式，详细见**官方手册18.7. Infrastructure for autotools-based packages**一节

```
$(eval $(generic-package))
```



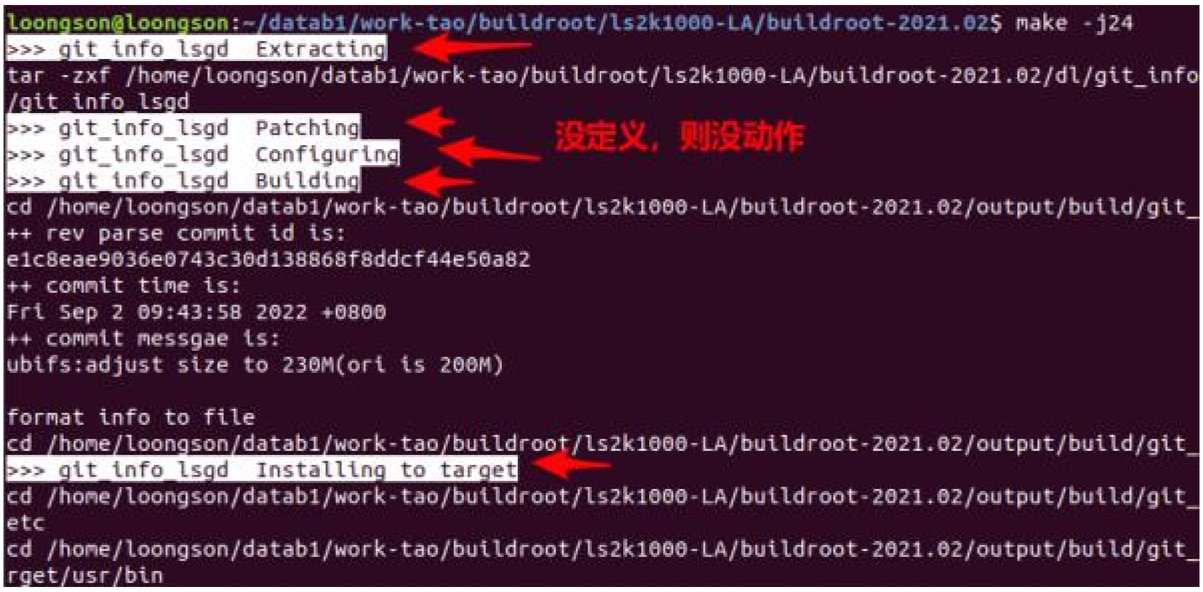
```
1  ##########################################################################
2  #
3  # git_info_lsgd
4  #
5  ##########################################################################
6
7  GIT_INFO_LSGD_SITE=$(TOPDIR)/dl/git_info_lsgd/git_info_lsgd.tar.gz
8  GIT_INFO_LSGD_SITE_METHOD=file
9
10 define GIT_INFO_LSGD_EXTRACT_CMDS
11         tar -zxf $(GIT_INFO_LSGD_SITE) -C $(@D)
12 endef
13
14 define GIT_INFO_LSGD_BUILD_CMDS
15         cd $(@D) && chmod a+x git-info-get.sh && ./git-info-get.sh
16         cd $(@D) && chmod a+x fs_git_info_lsgd
17 endef
18
19 define GIT_INFO_LSGD_INSTALL_TARGET_CMDS
20         cd $(@D) && cp git_info_ls $(TARGET_DIR)/etc
21         cd $(@D) && cp fs_git_info_lsgd $(TARGET_DIR)/usr/bin
22 endef
23
24 $(eval $(generic-package))
```

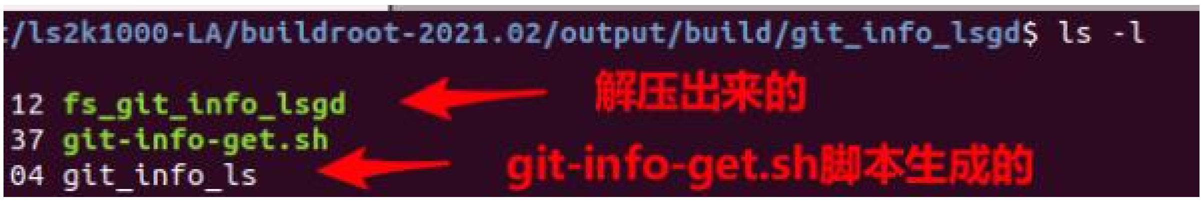
下面将以*git_info_lsgd*的编译来展示上述内容的
作用 输出如下：

```
>>> git_info_lsgd   Extracting
tar -zxf /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-
2021.02/dl/git_info_lsgd/git_info_lsgd.tar.gz -C /home/loongson/datab1/work-
tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_info_lsgd
>>>       git_info_lsgd   Patching
>>>       git_info_lsgd   Configuring
>>> git_info_lsgd         Building
```

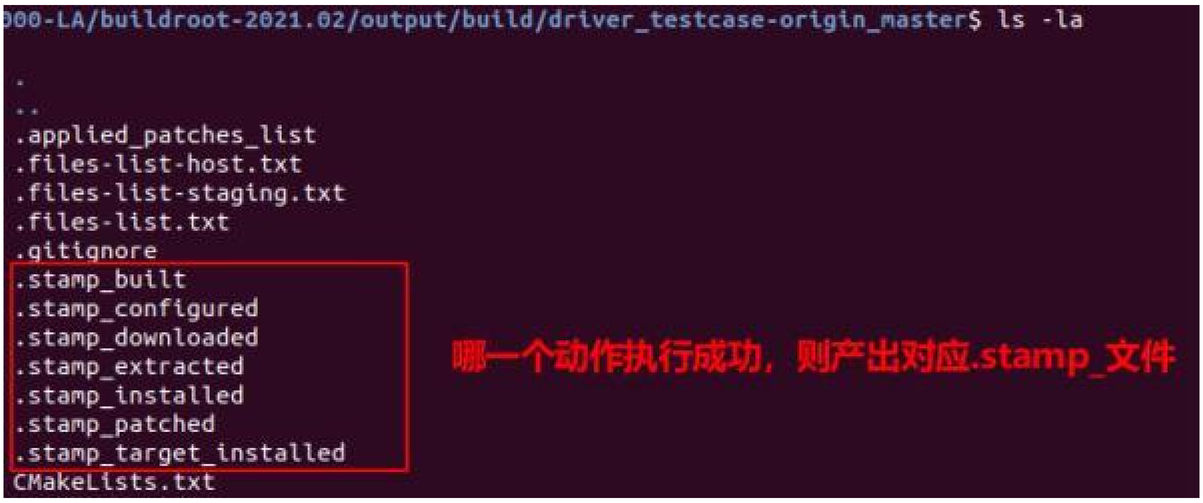
```
++ rev parse commit id is:
e1c8eae9036e0743c30d138868f8ddcf44e5
0a82 ++ commit time is:
Fri Sep 2 09:43:58 2022 +0800
++ commit messgae is:
ubifs:adjust size to 230M(ori is 200M)

format info to file
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-
2021.02/output/build/git_info_lsgd && chmod a+x fs_git_info_lsgd
>>> git_info_lsgd  Installing to target
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-
2021.02/output/build/git_info_lsgd && cp git_info_ls /home/loongson/datab1/work-
tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/target/etc
```

可见执行的动作和上述的内容是一一对应的


```
loongson@loongson:~/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02$ make -j24
>>> git_info_lsgd  Extracting
tar -zxf /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/dl/git_info
/git_info_lsgd
>>> git_info_lsgd  Patching
>>> git_info_lsgd  Configuring
>>> git_info_lsgd  Building
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_
++ rev parse commit id is:
e1c8eae9036e0743c30d138868f8ddcf44e50a82
++ commit time is:
Fri Sep 2 09:43:58 2022 +0800
++ commit messgae is:
ubifs:adjust size to 230M(ori is 200M)

format info to file
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_
>>> git_info_lsgd  Installing to target
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_
etc
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_
rget/usr/bin
```

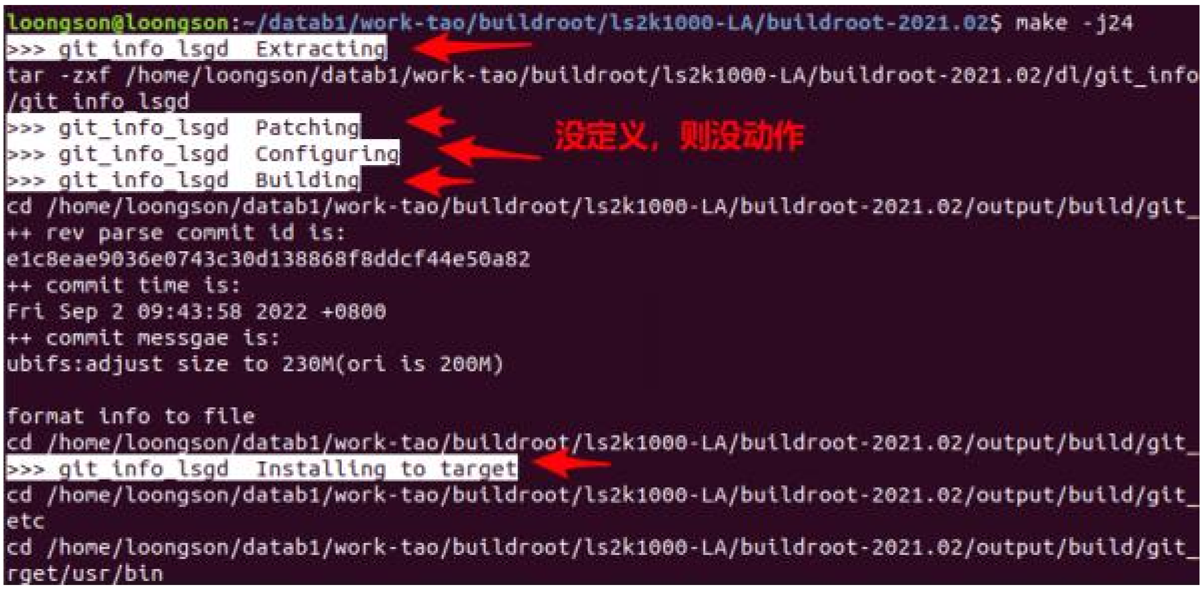

```
:/ls2k1000-LA/buildroot-2021.02/output/build/git_info_lsgd$ ls -l

12 fs_git_info_lsgd
37 git-info-get.sh
04 git_info_ls
```

见例子*driver_testcase*，*.stamp_xxx*文件是什么动作成功执行之后生成的标记，那么可以手动删除这些 标记，那么就会以那个动作开始，执行构建任务。

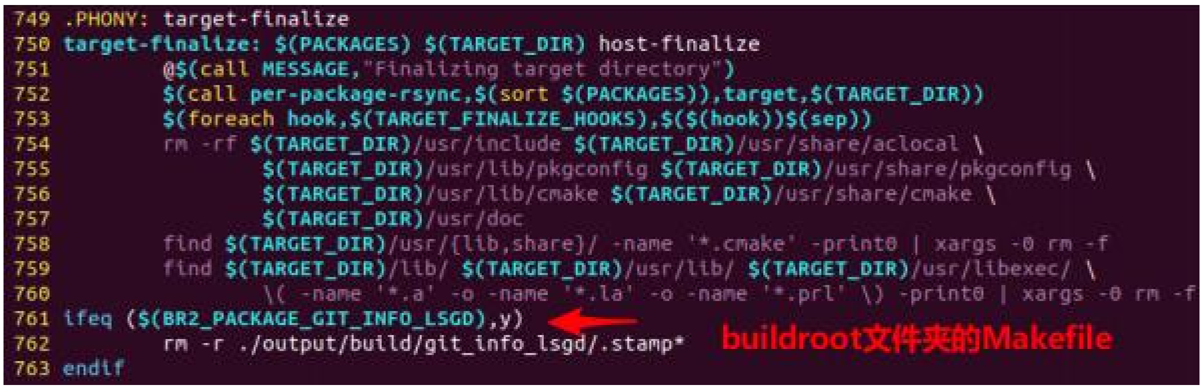


```
000-LA/buildroot-2021.02/output/build/driver_testcase-origin_master$ ls -la

.
..
.applied_patches_list
.files-list-host.txt
.files-list-staging.txt
.files-list.txt
.gitignore
.stamp_built
.stamp_configured
.stamp_downloaded
.stamp_extracted
.stamp_installed
.stamp_patched
.stamp_target_installed
CMakeLists.txt
```

所以在*git_info_lsgd*里面是找不到*.stamp_xxx*文件是正常的，当时设计的时候为了保证每次*make*的时候 都能记录信息，从而改了*buildroot*的一些底层逻辑。



```
loongson@loongson:~/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02$ make -j24
>>> git_info_lsgd  Extracting
tar -zxf /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/dl/git_info
/git_info_lsgd
>>> git_info_lsgd  Patching
>>> git_info_lsgd  Configuring
>>> git_info_lsgd  Building
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_
++ rev parse commit id is:
e1c8eae9036e0743c30d138868f8ddcf44e50a82
++ commit time is:
Fri Sep 2 09:43:58 2022 +0800
++ commit messgae is:
ubifs:adjust size to 230M(ori is 200M)

format info to file
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_
>>> git_info_lsgd  Installing to target
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_
etc
cd /home/loongson/datab1/work-tao/buildroot/ls2k1000-LA/buildroot-2021.02/output/build/git_
rget/usr/bin
```



```
.PHONY: target-finalize
target-finalize: $(PACKAGES) $(TARGET_DIR) host-finalize
	@$(call MESSAGE,"Finalizing target directory")
	$(call per-package-rsync,$(sort $(PACKAGES)),target,$(TARGET_DIR))
	$(foreach hook,$(TARGET_FINALIZE_HOOKS),$($(hook))$(sep))
	rm -rf $(TARGET_DIR)/usr/include $(TARGET_DIR)/usr/share/aclocal \
		$(TARGET_DIR)/usr/lib/pkgconfig $(TARGET_DIR)/usr/share/pkgconfig \
		$(TARGET_DIR)/usr/lib/cmake $(TARGET_DIR)/usr/share/cmake \
		$(TARGET_DIR)/usr/doc
	find $(TARGET_DIR)/usr/{lib,share}/ -name '*.cmake' -print0 | xargs -0 rm -f
	find $(TARGET_DIR)/lib/ $(TARGET_DIR)/usr/lib/ $(TARGET_DIR)/usr/libexec/ \
		\( -name '*.a' -o -name '*.la' -o -name '*.prl' \) -print0 | xargs -0 rm -f
ifeq ($(BR2_PACKAGE_GIT_INFO_LSGD),y)
	rm -r ./output/build/git_info_lsgd/.stamp*
endif
```

9.2.OpenWrt

*OpenWrt*是一个为嵌入式设备（通常是无线路由器）开发的高扩展度的*GNU/Linux*发行版。 与许多其他 路由器的发行版不同， *OpenWrt*是一个完全为嵌入式设备构建的功能全面、 易于修改的由现代*Linux*内 核驱动的操作系统。 在实践中，这意味着您可以得到您需要的所有功能，却仍能避免臃肿。

*OpenWrt*不是一个单一且不可更改的固件，而是提供了具有软件包管理功能的完全可写的文件系统，让您 通过使用适配任何应用的软件包来定制设备。 对于开发人员来说， *OpenWrt*是一个无需围绕它构建完整 固件就能开发应用程序的框架；对于普通用户来说，这意味着拥有了完全定制的能力，能以意想不到的方 式使用该设备。

*OpenWrt*官方网站：[*https://openwrt.org*](https://openwrt.org)

OpenWrt官方Git仓库：https://github.com/openwrt/openwrt

9.2.1.OpenWrt 编译

从BSP 包的**文件系统**目录下找到OpenWrt 的源码并解码。

```
$ cp configs/loongson_2k300_config .config


$ make -j24
```

可以选择 V=sc 打印编译 log:

```
$ make V=sc -j24
```

编译成功之后会在 `bin/target/loongson/ls2k300`下生成`openwrt-loongson-ls2k500-loongson_gd_ls2k500_mini-ubifs-root.ubi`文件，将其改名为 `rootfs-ubifs-ze.img`后烧录到板 卡即可
。

直接复制保存config

```
$ cp .config .configs/xxx_config
```

注意：
./scripts/feeds 已固化，不需要再运行以下命令：

```
$ ./scripts/feeds update -a $ ./
scripts/feeds install -a
```

9.2.2.OpenWrt 二次开发

9.2.2.1.OpenWrt 网络定制，修改 LAN IP

修改宏 LOONGSON_LAN_IPADDR 即可，如果遇到复杂的网络定制需求，可通过以下步骤排查实现

1. openwrt 网络配置由 `/etc/config/network` 决定
2. `/etc/config/network` 在系统第一次运行时由 `/etc/board.d/02_network` 与 `/bin/config_generate` 相互作用后生成，在定制时可以通过修改这两个文件达成目标
3. `/etc/board.d/02_network` 中 `ucidef_set_interface_lan`会指定 lan 口，`ucidef_set_interface_wan`指定 wan 口， `ucidef_set_interfaces_lan_wan`同时指定 lan wan 口
4. `/bin/config_generate` 中 `generate_network`函数可指定 IP

9.2.2.2.OpenWrt 添加软件包

在运行 `./scripts/feeds update -a ./scripts/feeds install -a` 后，绝大多数软件包都能在 `menuconfig`中找到
目前尚未新增不在 feeds 中的软件

9.3. 系统镜像包二次开发

这一节主要介绍的是根据广东龙芯提供的系统镜像包进行二次开发的简单操作指引。

二次开发通常指的是在系统镜像进行中加入用户自己的软件和服务， **这样只需要烧录自制的镜像，就可以烧录后直接上线使用，无需再次部署。**

系统镜像包通常包括：

- rootfs.tar.gz
- rootfs.img

以上的文件都有广东龙芯方提供的原始版本。通常是以下三个文件一起提供的。

- rootfs.tar.gz
- uImage
- rootfs.img

rootfs.img 是由 rootfs.tar.gz 和 uImage 制作而来的。

也就是说使用 rootfs.tar.gz 、 uImage、 ramdisk.gz 安装系统和直接使用 rootfs.img 烧录系统， 最后得 到的系统基本是一样的。

所以二次开发的流程为：

1. 解压 rootfs.tar.gz
2. 修改解压后的文件夹
3. 重新打包 rootfs.tar.gz
4. 利用新的 rootfs.tar.gz 和 uImage 制作 rootfs.img

为了方便以上流程。广东龙芯方会提供一个 system_package_img_generate.tar.gz 的包。在 ubuntu(x86) 或者 loongnix(龙芯台式机) 下解压后会得到一份 README.md 和三份脚本。

利用这三份脚本，就能完成上述的第1、3、4步

。 README.md 会介绍该工具的使用方式。

本文也会简单地介绍此用法。

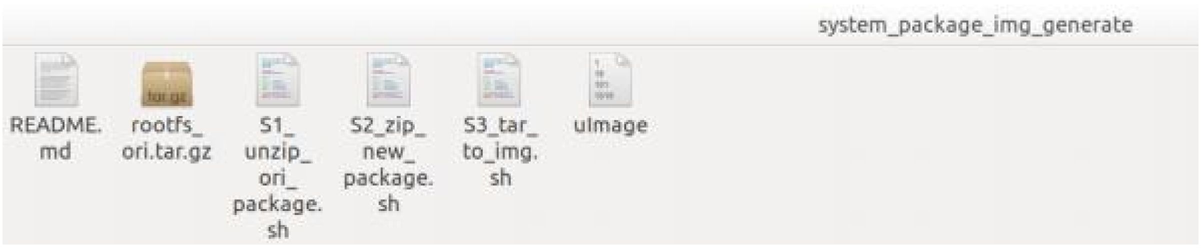
首先解压 system_package_img_generate.tar.gz，然后进入 system_package_img_generate 文件夹。

```
tar -zxf system_package_img_generate.tar.gz cd

system_package_img_generate
```

把广东龙芯方提供的 rootfs .tar .gz 和 uImage 放到

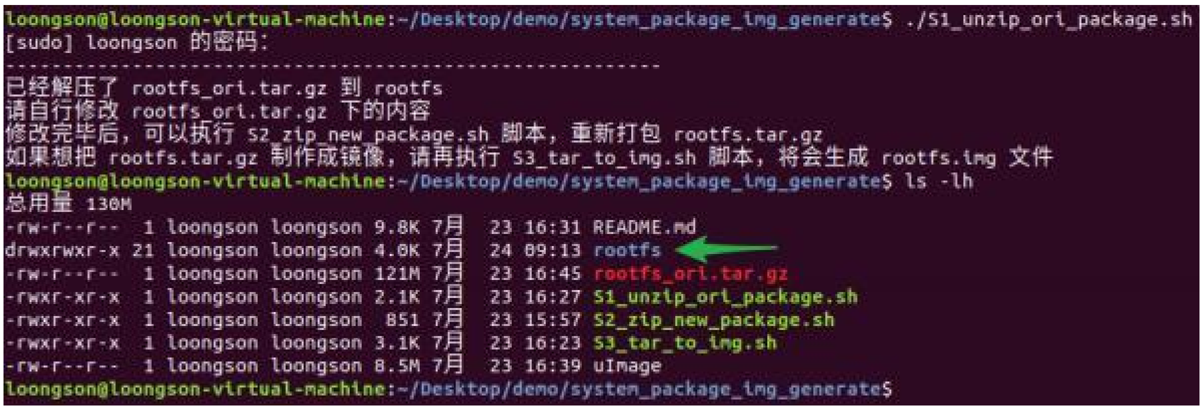
system_package_img_generate 下。 把 rootfs.tar.gz 改名为 rootfs_ori

.tar .gz。



解压 rootfs_ori.tar.gz，执行以下命令：

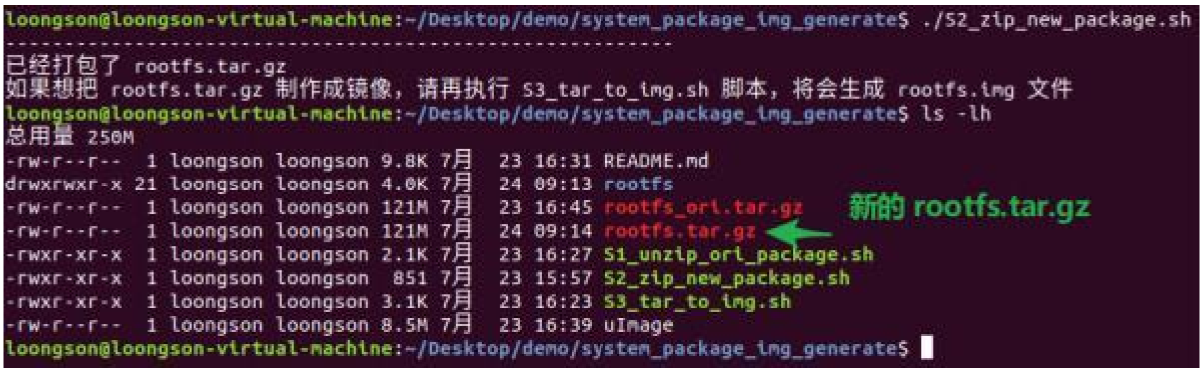
```
./S1_unzip_ori_package.sh
```


```
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$ ./S1_unzip_ori_package.sh
[sudo] loongson 的密码:
----------------------------------------------------------
已经解压了 rootfs_ori.tar.gz 到 rootfs
请自行修改 rootfs_ori.tar.gz 下的内容
修改完毕后, 可以执行 S2_zip_new_package.sh 脚本, 重新打包 rootfs.tar.gz
如果想把 rootfs.tar.gz 制作成镜像, 请再执行 S3_tar_to_img.sh 脚本, 将会生成 rootfs.img 文件
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$ ls -lh
总用量 130M
-rw-r--r--  1 loongson loongson 9.8K 7月  23 16:31 README.md
drwxrwxr-x 21 loongson loongson 4.0K 7月  24 09:13 rootfs
-rw-r--r--  1 loongson loongson 121M 7月  23 16:45 rootfs_ori.tar.gz
-rwxr-xr-x  1 loongson loongson 2.1K 7月  23 16:27 S1_unzip_ori_package.sh
-rwxr-xr-x  1 loongson loongson  851 7月  23 15:57 S2_zip_new_package.sh
-rwxr-xr-x  1 loongson loongson 3.1K 7月  23 16:23 S3_tar_to_img.sh
-rw-r--r--  1 loongson loongson 8.5M 7月  23 16:39 uImage
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$
```

随后在得到的 *rootfs* 文件夹内进行修改，进行二次开

发。 接着重新打包 *rootfs.tar.gz* 。

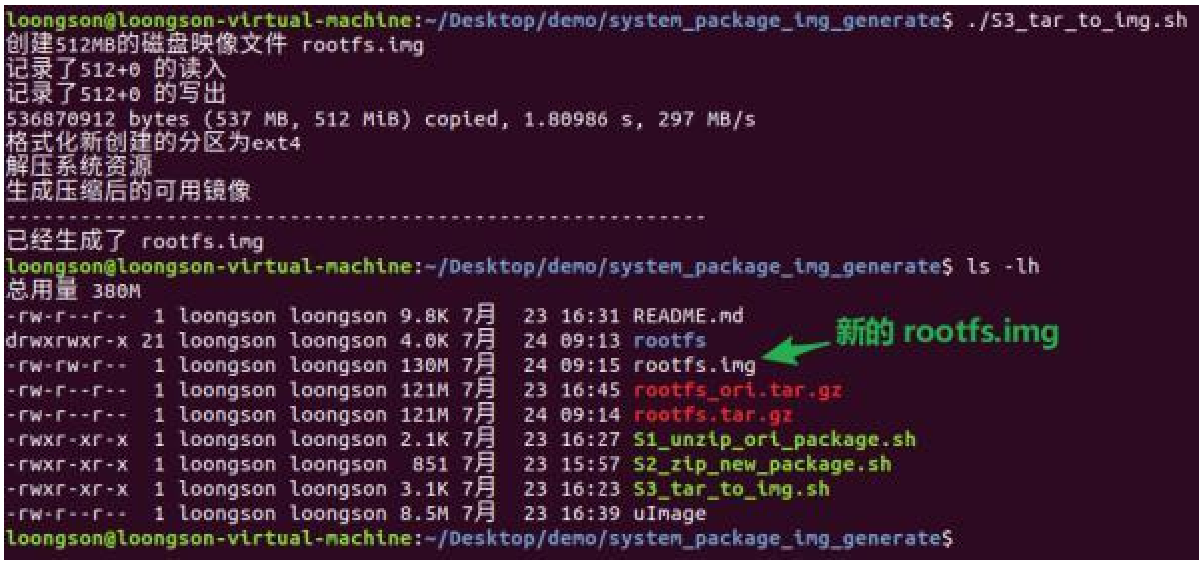
```
./S2_zip_new_package.sh
```


```
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$ ./S2_zip_new_package.sh
----------------------------------------------------------
已经打包了 rootfs.tar.gz
如果想把 rootfs.tar.gz 制作成镜像, 请再执行 S3_tar_to_img.sh 脚本, 将会生成 rootfs.img 文件
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$ ls -lh
总用量 250M
-rw-r--r--  1 loongson loongson 9.8K 7月  23 16:31 README.md
drwxrwxr-x 21 loongson loongson 4.0K 7月  24 09:13 rootfs
-rw-r--r--  1 loongson loongson 121M 7月  23 16:45 rootfs_ori.tar.gz
-rw-r--r--  1 loongson loongson 121M 7月  24 09:14 rootfs.tar.gz
-rwxr-xr-x  1 loongson loongson 2.1K 7月  23 16:27 S1_unzip_ori_package.sh
-rwxr-xr-x  1 loongson loongson  851 7月  23 15:57 S2_zip_new_package.sh
-rwxr-xr-x  1 loongson loongson 3.1K 7月  23 16:23 S3_tar_to_img.sh
-rw-r--r--  1 loongson loongson 8.5M 7月  23 16:39 uImage
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$
```

这样就会得到新的 *rootfs.tar.gz* 文件。

如果目标是为了 *rootfs.img*，那么请执行以下命令：

```
./S3_tar_to_img.sh
```


```
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$ ./S3_tar_to_img.sh
创建512MB的磁盘映像文件 rootfs.img
记录了512+0 的读入
记录了512+0 的写出
536870912 bytes (537 MB, 512 MiB) copied, 1.80986 s, 297 MB/s
格式化新创建的分区为ext4
解压系统资源
生成压缩后的可用镜像
----------------------------------------------------------
已经生成了 rootfs.img
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$ ls -lh
总用量 380M
-rw-r--r--  1 loongson loongson 9.8K 7月  23 16:31 README.md
drwxrwxr-x 21 loongson loongson 4.0K 7月  24 09:13 rootfs
-rw-rw-r--  1 loongson loongson 130M 7月  24 09:15 rootfs.img
-rw-r--r--  1 loongson loongson 121M 7月  23 16:45 rootfs_ori.tar.gz
-rw-r--r--  1 loongson loongson 121M 7月  24 09:14 rootfs.tar.gz
-rwxr-xr-x  1 loongson loongson 2.1K 7月  23 16:27 S1_unzip_ori_package.sh
-rwxr-xr-x  1 loongson loongson  851 7月  23 15:57 S2_zip_new_package.sh
-rwxr-xr-x  1 loongson loongson 3.1K 7月  23 16:23 S3_tar_to_img.sh
-rw-r--r--  1 loongson loongson 8.5M 7月  23 16:39 uImage
loongson@loongson-virtual-machine:~/Desktop/demo/system_package_img_generate$
```

详细的说明可以看 *system_package_img_generate .tar .gz* 中的README.md文件。

9.4. uboot 写入镜像文件

将全盘镜像直接写入MMC/SSD，即可完成安装/更新

```
=> tftp disk.img

=> mmc write ${loadaddr} 0 40000
```

查看系统分区

```
=> mmc part
Part    Start Sector    Num Sectors    UUID            Type
  1     8192            12615680       00000000-01     83
```

9.5. 进入系统后扩大分区与文件系统

制作的全盘镜像比实际使用的EMMC容量要小，系统启动后可执行以下命令扩大分区

```
parted /dev/mmcblk0 --script -- resizepart 1 -0

resize2fs /dev/mmcblk0p1
```

十、 linux定制与裁减

10.0.内核特性

基于linux-5.10.0 移植开发

- 支持YAFFS2/CRAMFS/NFS/UBIFS/NFS/FAT32

等格式的文件系统 • 支持SLC NAND Flash驱动，容量：

256MB/512MB/1GB

- 支持看门狗驱

动 • 支持RTC驱

动

- 支持LED驱

动 • 支持按键

驱动

- 支持SPI驱动:最大支持4个片选
- 支持I2C驱动:支持100KHz，

400KHz • 支持PWM驱动

- 支持 USB Host驱动
- 支持串口:最大支持12路串口，波特率稳定支持460800
- 支持GMAC千兆以太网驱动：支持RTL8211E，支持YT8521
- 支持显示驱动:RGB接口可通过芯片转接LVDS、VGA、 HDMI、 DVI，支持

双屏显示 · 支持LCD背光驱动:gpio控制或pwm控制

· 支持PCIE控制器驱动:支持usb 3.0 pcie卡，sata 3.0 pcie卡，千兆网卡等

- 支持CAN驱:最高波特率

1MHz • 支持GPIO驱动

- 支持PINCTRL驱动:用于引脚复用

配置 • 支持CLK驱动

- 支持硬件随机数生成器
- 支持温度监测:用于读取芯片温度

10.1.内核编译

10.1.1.内核编译流程

```
设置交叉工具链等环境
$ source ./set_env.sh
====>setup env for LoongArch... 指
定板卡配置
$ make loongson_2k300_defconfig
make[1]: Entering directory '/home/vm/kernel-5.10-LoongArch'
  GEN     Makefile
#
# No change to .config #
make[1]: Leaving directory '/home/vm/kernel-5.10-LoongArch' 开始
编译
$ make uImage
  GEN     .version
  CHK     include/generated/compile.h
  UPD     include/generated/compile.h
```

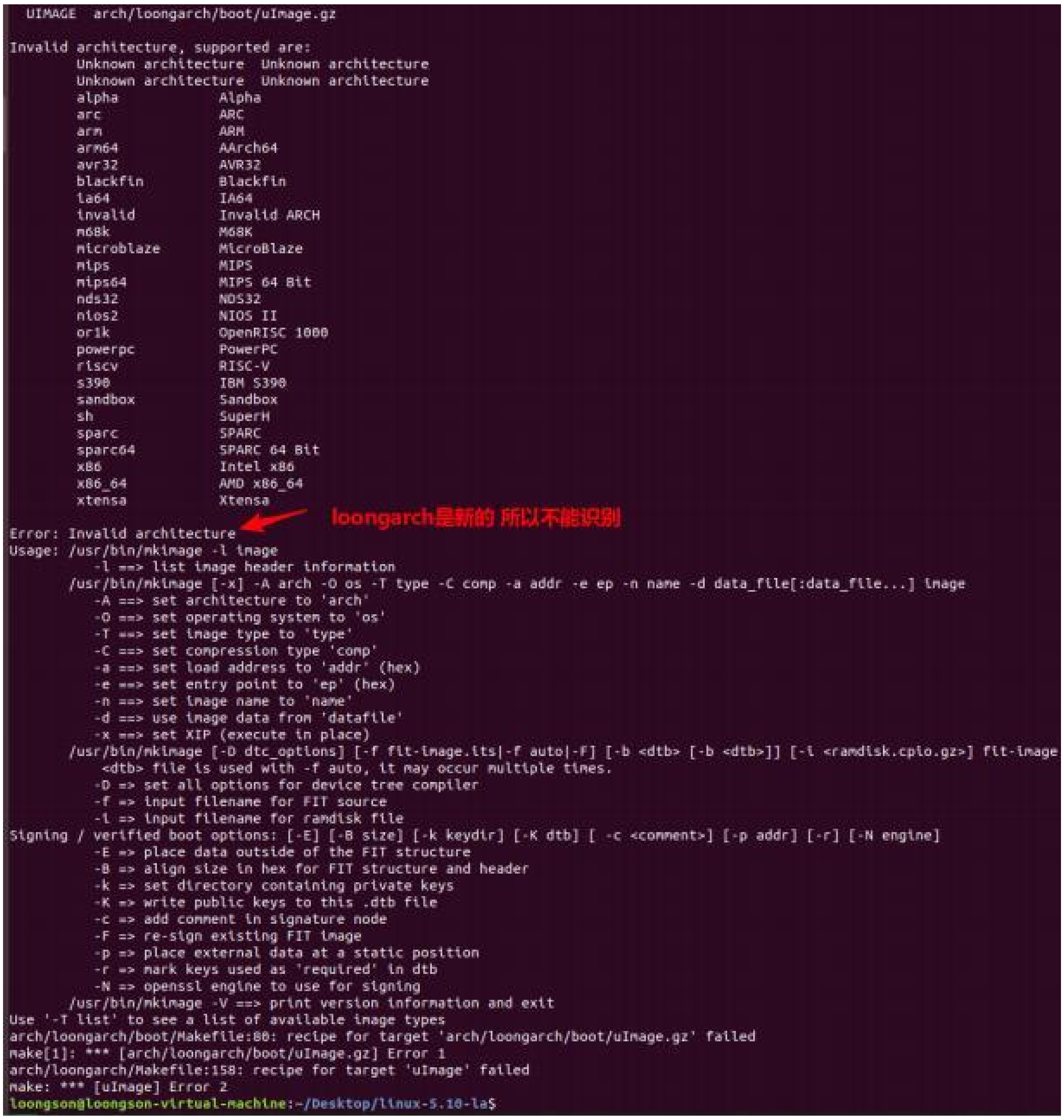
```
CC      init/version.o   AR
init/built-in.a
```

```
  LD      vmlinux.o
  MODPOST vmlinux.symvers
  MODINFO modules.builtin.modinfo
  GEN     modules.builtin
  LD      .tmp_vmlinux.kallsyms1
  KSYMS   .tmp_vmlinux.kallsyms1.S
  AS      .tmp_vmlinux.kallsyms1.S LD
  .tmp_vmlinux.kallsyms2
  KSYMS   .tmp_vmlinux.kallsyms2.S
  AS      .tmp_vmlinux.kallsyms2.S
  LD      vmlinux
  SORTTAB vmlinux
  SYSMAP  System.map
  OBJCOPY arch/loongarch/boot/vmlinux.bin
  GZIP    arch/loongarch/boot/vmlinux.bin.gz
  UIMAGE  arch/loongarch/boot/uImage.gz
Image Name:   Linux-5.10.0.lsgd-g217c34b2e4e3
Created:      Sat Sep 17 14:20:28 2022
```

编译成功之后会在 `arch/loongarch/boot/`生成`uImage`文件。

10.1.2.编译loongarch内核的问题修复

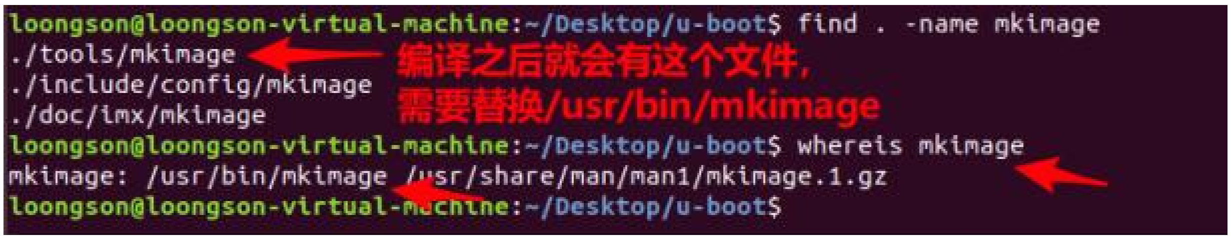
1. 内核编译出现 invalid architecture



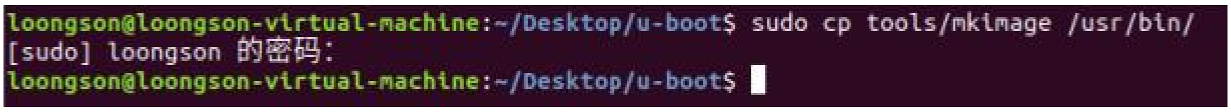
```
  UIMAGE  arch/loongarch/boot/uImage.gz

Invalid architecture, supported are:
        Unknown architecture  Unknown architecture
        Unknown architecture  Unknown architecture
        alpha             Alpha
        arc               ARC
        arm               ARM
        arm64             AArch64
        avr32             AVR32
        blackfin          Blackfin
        ia64              IA64
        invalid           Invalid ARCH
        m68k              M68K
        microblaze        MicroBlaze
        mips              MIPS
        mips64            MIPS 64 Bit
        nds32             NDS32
        nios2             NIOS II
        or1k              OpenRISC 1000
        powerpc           PowerPC
        riscv             RISC-V
        s390              IBM S390
        sandbox           Sandbox
        sh                SuperH
        sparc             SPARC
        sparc64           SPARC 64 Bit
        x86               Intel x86
        x86_64            AMD x86_64
        xtensa            Xtensa
Error: Invalid architecture
Usage: /usr/bin/mkimage -l image
          -l ==> list image header information
       /usr/bin/mkimage [-x] -A arch -O os -T type -C comp -a addr -e ep -n name -d data_file[:data_file...] image
          -A ==> set architecture to 'arch'
          -O ==> set operating system to 'os'
          -T ==> set image type to 'type'
          -C ==> set compression type 'comp'
          -a ==> set load address to 'addr' (hex)
          -e ==> set entry point to 'ep' (hex)
          -n ==> set image name to 'name'
          -d ==> use image data from 'datafile'
          -x ==> set XIP (execute in place)
       /usr/bin/mkimage [-D dtc_options] [-f fit-image.its|-f auto|-F] [-b <dtb> [-b <dtb>]] [-i <ramdisk.cpio.gz>] fit-image
           <dtb> file is used with -f auto, it may occur multiple times.
          -D => set all options for device tree compiler
          -f => input filename for FIT source
          -i => input filename for ramdisk file
Signing / verified boot options: [-E] [-B size] [-k keydir] [-K dtb] [ -c <comment>] [-p addr] [-r] [-N engine]
          -E => place data outside of the FIT structure
          -B => align size in hex for FIT structure and header
          -k => set directory containing private keys
          -K => write public keys to this .dtb file
          -c => add comment in signature node
          -F => re-sign existing FIT image
          -p => place external data at a static position
          -r => mark keys used as 'required' in dtb
          -N => openssl engine to use for signing
       /usr/bin/mkimage -V ==> print version information and exit
Use '-T list' to see a list of available image types
arch/loongarch/boot/Makefile:80: recipe for target 'arch/loongarch/boot/uImage.gz' failed
make[1]: *** [arch/loongarch/boot/uImage.gz] Error 1
arch/loongarch/Makefile:158: recipe for target 'uImage' failed
make: *** [uImage] Error 2
loongson@loongson-virtual-machine:~/Desktop/linux-5.10-la$
```

原因是*loongarch*是新的架构， *u-boot-tools*包里面的*mkimage*不能识别此架构。所以无法制作*uImage.gz*。

要解决这个问题就要到*uboot*源码里面，编译*uboot*（参考 *11.1.如何编译uboot*）。编译完成后在 *uboot* 的 *tools* 目录下会生成 *mkimage* 程序，将该程序替换系统 */usr/bin* 下的 *mkimage*。

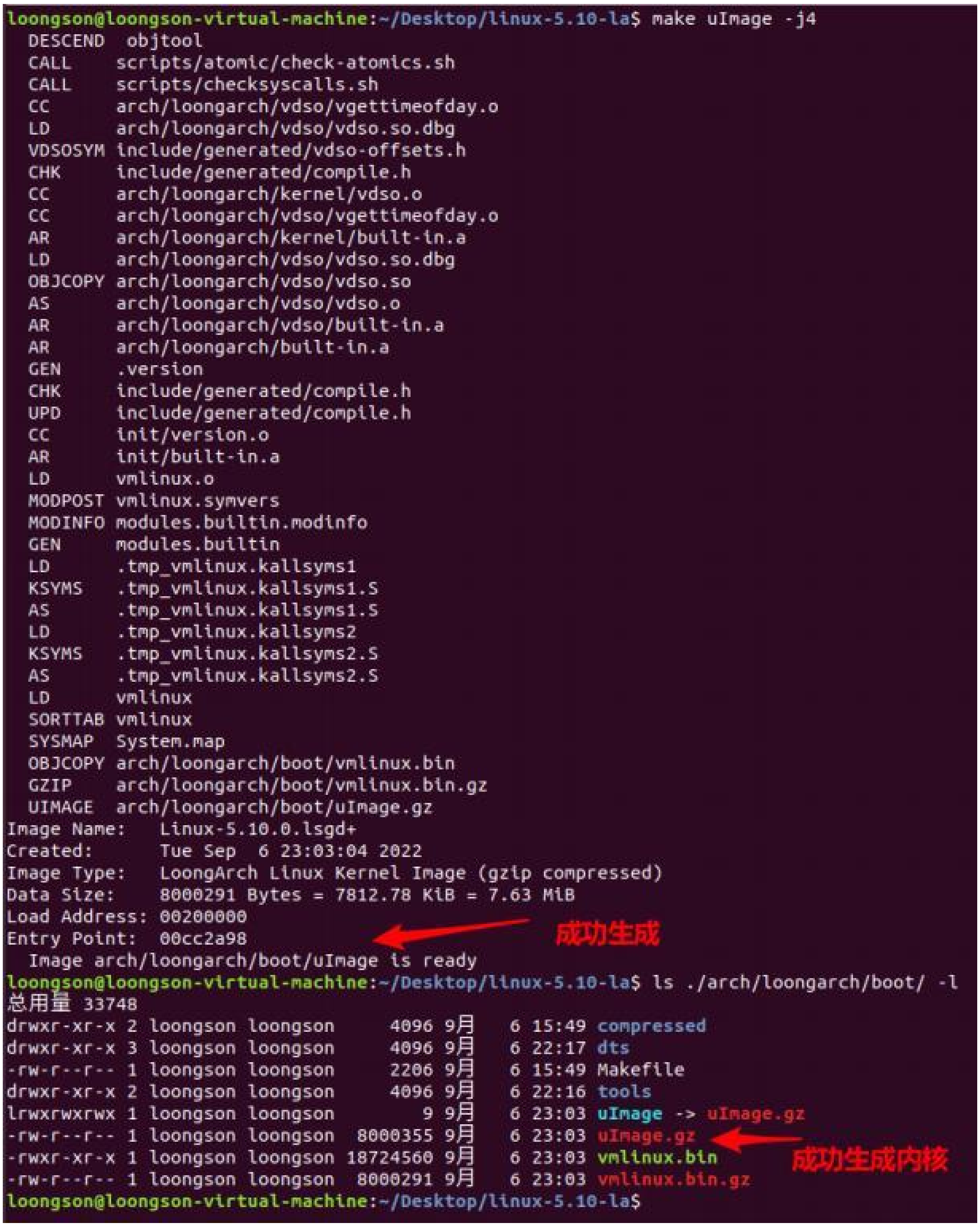


```
loongson@loongson-virtual-machine:~/Desktop/u-boot$ find . -name mkimage
./tools/mkimage
./include/config/mkimage
./doc/imx/mkimage
loongson@loongson-virtual-machine:~/Desktop/u-boot$ whereis mkimage
mkimage: /usr/bin/mkimage /usr/share/man/man1/mkimage.1.gz
loongson@loongson-virtual-machine:~/Desktop/u-boot$
```



```
loongson@loongson-virtual-machine:~/Desktop/u-boot$ sudo cp tools/mkimage /usr/bin/
[sudo] loongson 的密码:
loongson@loongson-virtual-machine:~/Desktop/u-boot$
```

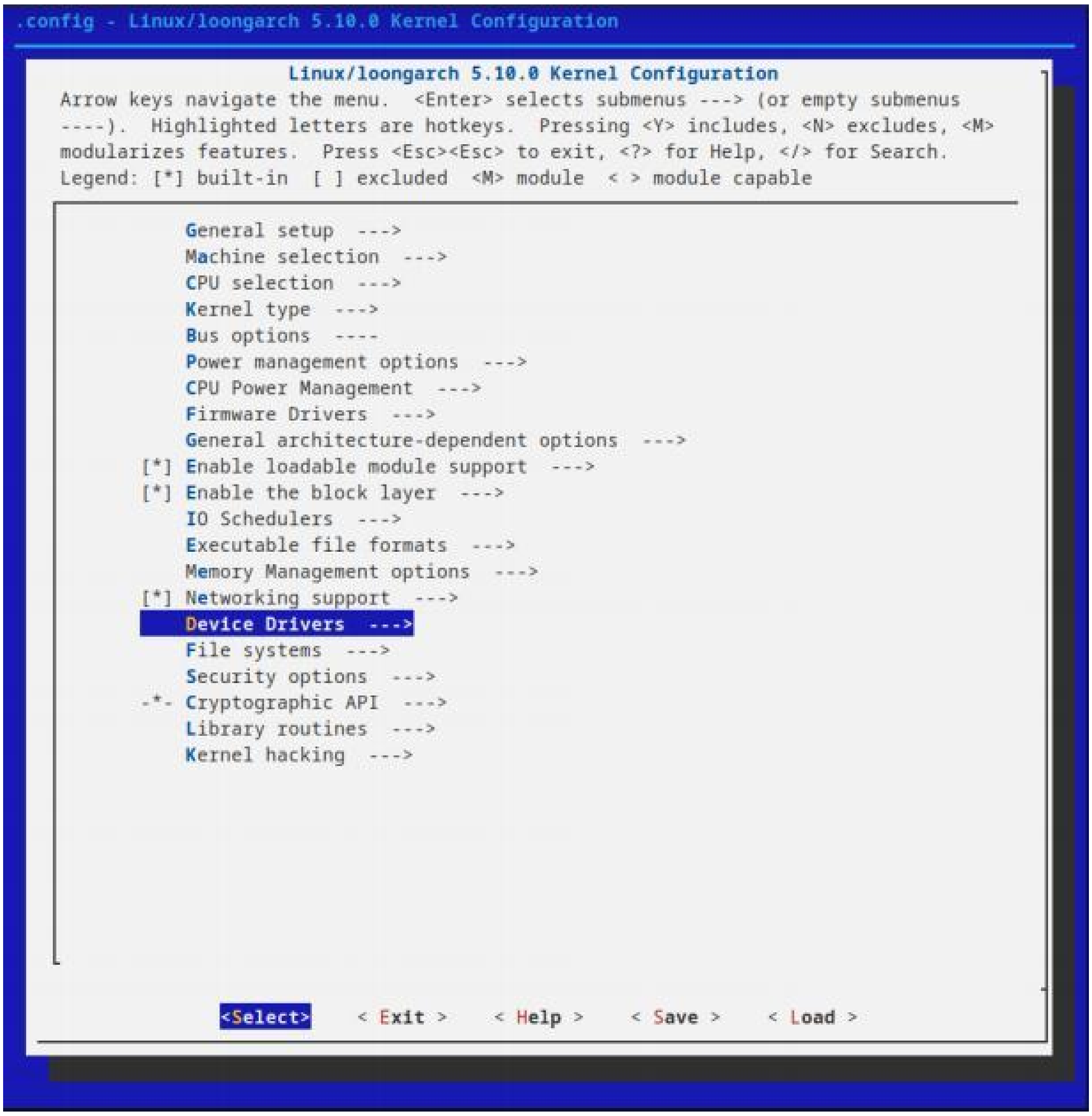
再次编译内核，就能成功得到*uImage.gz*文件



```
loongson@loongson-virtual-machine:~/Desktop/linux-5.10-la$ make uImage -j4
  DESCEND  objtool
  CALL    scripts/atomic/check-atomics.sh
  CALL    scripts/checksyscalls.sh
  CC      arch/loongarch/vdso/vgettimeofday.o
  LD      arch/loongarch/vdso/vdso.so.dbg
  VDSOSYM include/generated/vdso-offsets.h
  CHK     include/generated/compile.h
  CC      arch/loongarch/kernel/vdso.o
  CC      arch/loongarch/vdso/vgettimeofday.o
  AR      arch/loongarch/kernel/built-in.a
  LD      arch/loongarch/vdso/vdso.so.dbg
  OBJCOPY arch/loongarch/vdso/vdso.so
  AS      arch/loongarch/vdso/vdso.o
  AR      arch/loongarch/vdso/built-in.a
  AR      arch/loongarch/built-in.a
  GEN     .version
  CHK     include/generated/compile.h
  UPD     include/generated/compile.h
  CC      init/version.o
  AR      init/built-in.a
  LD      vmlinux.o
  MODPOST vmlinux.symvers
  MODINFO modules.builtin.modinfo
  GEN     modules.builtin
  LD      .tmp_vmlinux.kallsyms1
  KSYMS   .tmp_vmlinux.kallsyms1.S
  AS      .tmp_vmlinux.kallsyms1.S
  LD      .tmp_vmlinux.kallsyms2
  KSYMS   .tmp_vmlinux.kallsyms2.S
  AS      .tmp_vmlinux.kallsyms2.S
  LD      vmlinux
  SORTTAB vmlinux
  SYSMAP  System.map
  OBJCOPY arch/loongarch/boot/vmlinux.bin
  GZIP    arch/loongarch/boot/vmlinux.bin.gz
  UIMAGE  arch/loongarch/boot/uImage.gz
Image Name:   Linux-5.10.0.lsgd+
Created:      Tue Sep  6 23:03:04 2022
Image Type:   LoongArch Linux Kernel Image (gzip compressed)
Data Size:    8000291 Bytes = 7812.78 KiB = 7.63 MiB
Load Address: 00200000
Entry Point:  00cc2a98
  Image arch/loongarch/boot/uImage is ready
loongson@loongson-virtual-machine:~/Desktop/linux-5.10-la$ ls ./arch/loongarch/boot/ -l
总用量 33748
drwxr-xr-x 2 loongson loongson     4096 9月   6 15:49 compressed
drwxr-xr-x 3 loongson loongson     4096 9月   6 22:17 dts
-rw-r--r-- 1 loongson loongson     2206 9月   6 15:49 Makefile
drwxr-xr-x 2 loongson loongson     4096 9月   6 22:16 tools
lrwxrwxrwx 1 loongson loongson        9 9月   6 23:03 uImage -> uImage.gz
-rw-r--r-- 1 loongson loongson  8000355 9月   6 23:03 uImage.gz
-rwxr-xr-x 1 loongson loongson 18724560 9月   6 23:03 vmlinux.bin
-rw-r--r-- 1 loongson loongson  8000291 9月   6 23:03 vmlinux.bin.gz
loongson@loongson-virtual-machine:~/Desktop/linux-5.10-la$
```

10.2.驱动配置

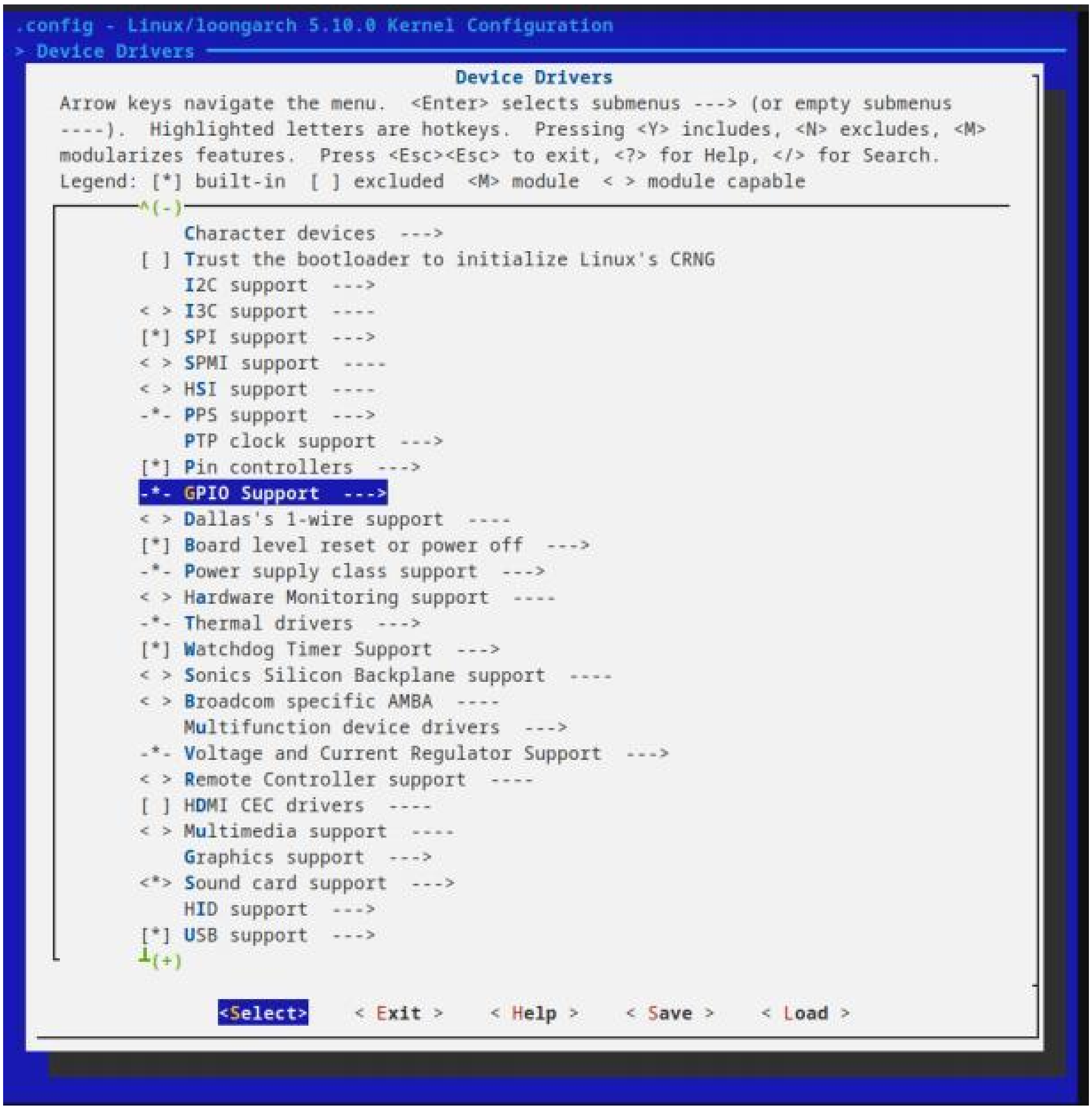
在主菜单页面中，选择"Device Drivers"选项，按回车进入。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
                      Linux/loongarch 5.10.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

            General setup  --->
            Machine selection  --->
            CPU selection  --->
            Kernel type  --->
            Bus options  ----
            Power management options  --->
            CPU Power Management  --->
            Firmware Drivers  --->
            General architecture-dependent options  --->
        [*] Enable loadable module support  --->
        [*] Enable the block layer  --->
            IO Schedulers  --->
            Executable file formats  --->
            Memory Management options  --->
        [*] Networking support  --->
            Device Drivers  --->
            File systems  --->
            Security options  --->
        -*- Cryptographic API  --->
            Library routines  --->
            Kernel hacking  --->

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

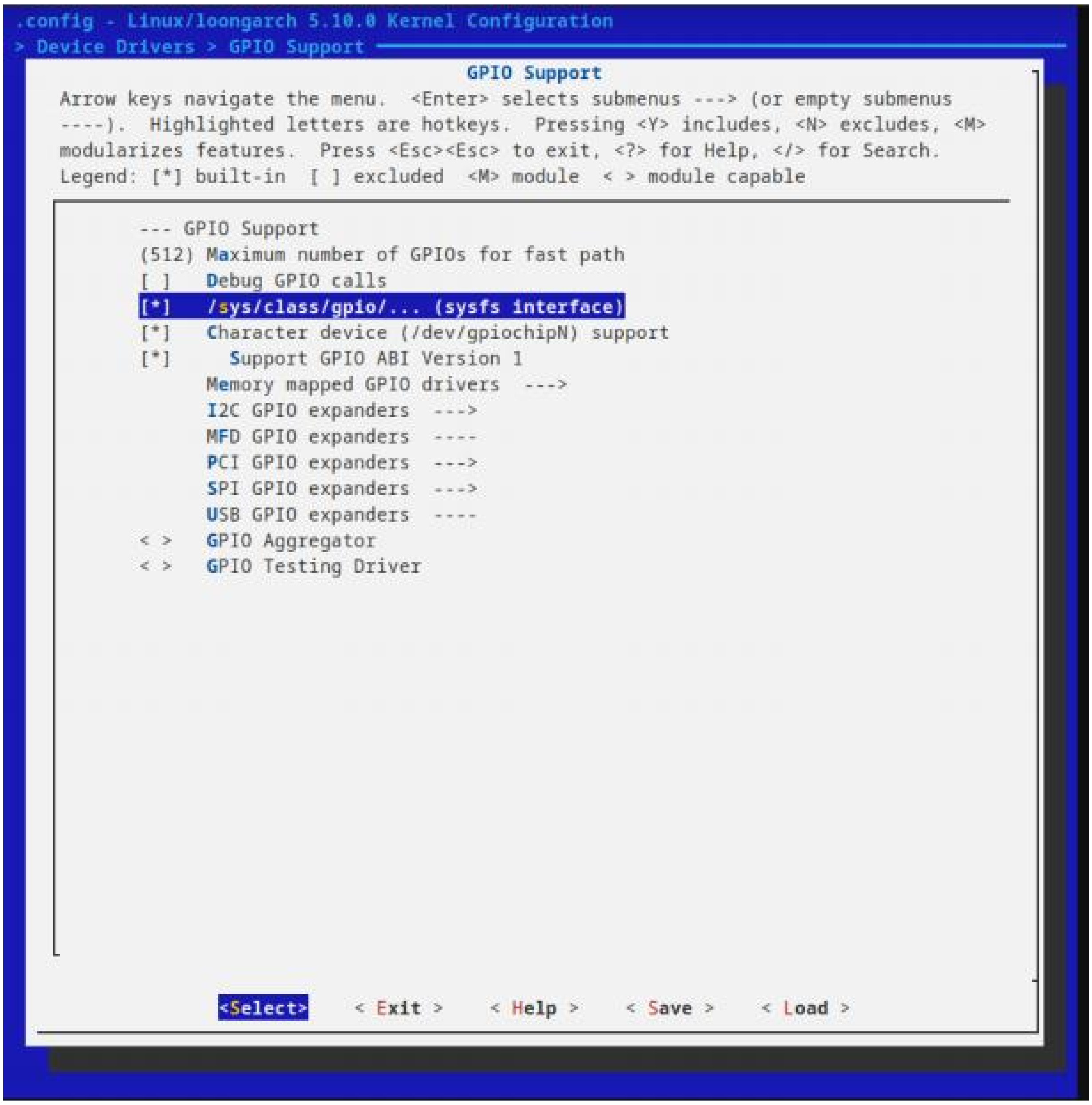
10.2.1.GPIO

Device Driver 界面选择“GPIO Support”选项，按回车进入。

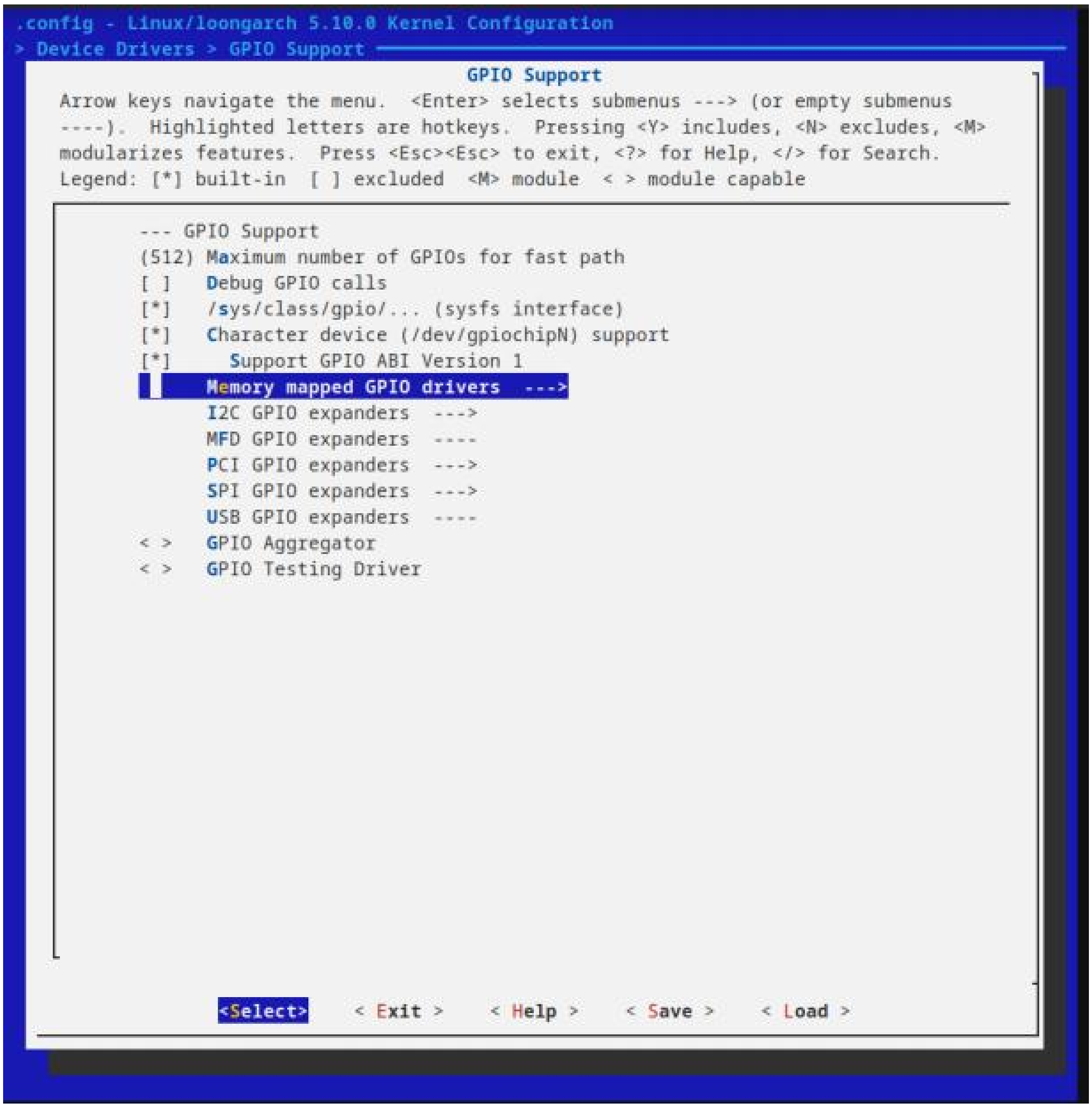


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers ─────────────────────────────────────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
 ┌───────^(-)───────────────────────────────────────────────────────────────
 │          Character devices  --->
 │       [ ] Trust the bootloader to initialize Linux's CRNG
 │           I2C support  --->
 │       < > I3C support  ----
 │       [*] SPI support  --->
 │       < > SPMI support  ----
 │       < > HSI support  ----
 │       -*- PPS support  --->
 │           PTP clock support  --->
 │       [*] Pin controllers  --->
 │       -*- GPIO Support  --->
 │       < > Dallas's 1-wire support  ----
 │       [*] Board level reset or power off  --->
 │       -*- Power supply class support  --->
 │       < > Hardware Monitoring support  ----
 │       -*- Thermal drivers  --->
 │       [*] Watchdog Timer Support  --->
 │       < > Sonics Silicon Backplane support  ----
 │       < > Broadcom specific AMBA  ----
 │           Multifunction device drivers  --->
 │       -*- Voltage and Current Regulator Support  --->
 │       < > Remote Controller support  ----
 │       [ ] HDMI CEC drivers  ----
 │       < > Multimedia support  ----
 │           Graphics support  --->
 │       <*> Sound card support  --->
 │           HID support  --->
 │       [*] USB support  --->
 └───────┴(+)

         <Select>    < Exit >    < Help >    < Save >    < Load >
```

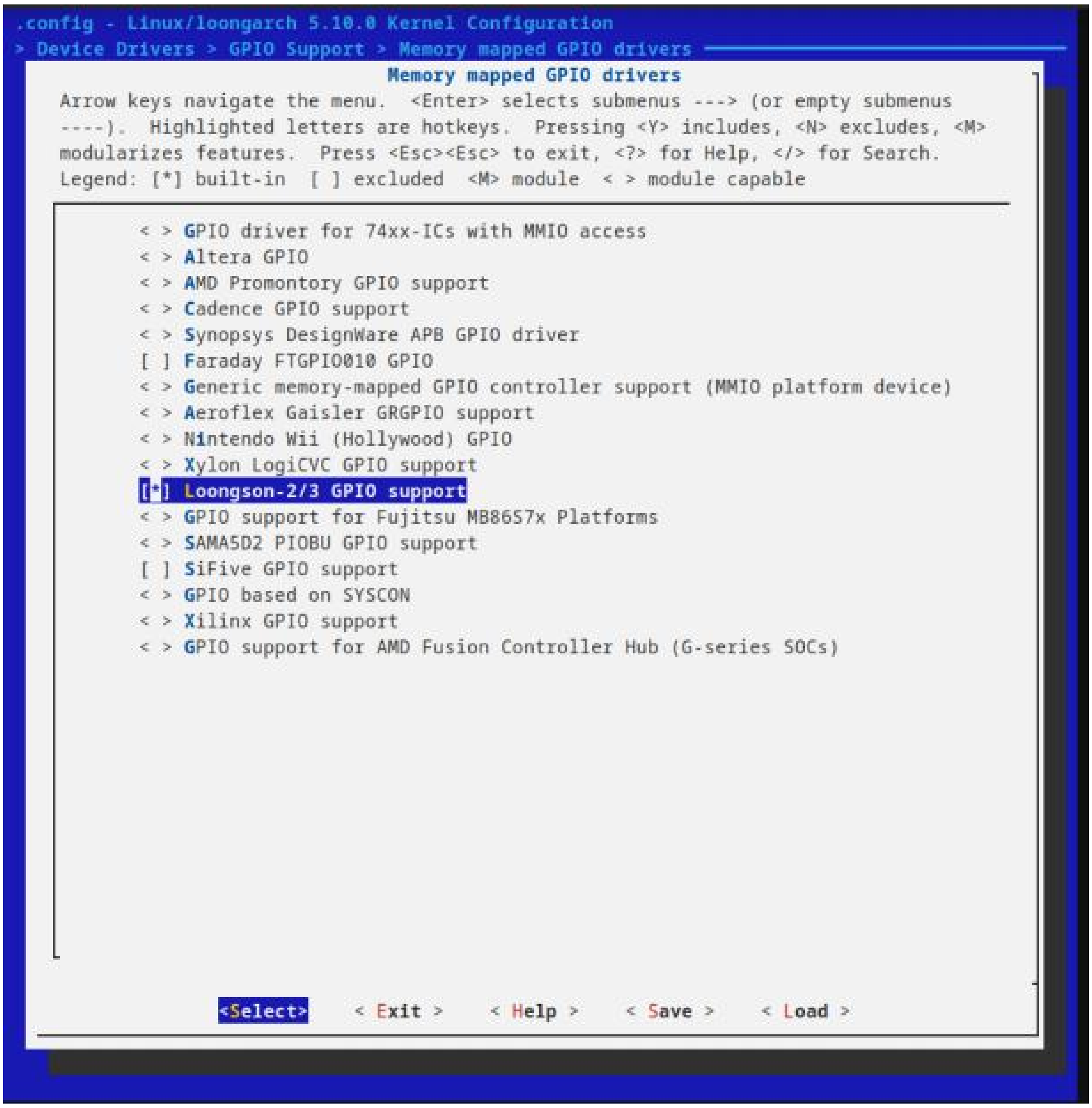
选项"/sys/class/gpio/…"选中的话，那么系统启动后，可以在/sys/class/gpio/文件夹下对gpio口进行操作
。


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > GPIO Support
                              GPIO Support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- GPIO Support
        (512) Maximum number of GPIOs for fast path
        [ ]   Debug GPIO calls
        [*]   /sys/class/gpio/... (sysfs interface)
        [*]   Character device (/dev/gpiochipN) support
        [*]     Support GPIO ABI Version 1
              Memory mapped GPIO drivers  --->
              I2C GPIO expanders  --->
              MFD GPIO expanders  ----
              PCI GPIO expanders  --->
              SPI GPIO expanders  --->
              USB GPIO expanders  ----
        < >   GPIO Aggregator
        < >   GPIO Testing Driver

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中 “ Memory mapped GPIO drivers” 选项， 并且回车进入。然后选中 ” loongson-2/3 GPIO support” 选项， 按空格选中 为“*”。即可使能龙芯2K1000自带的gpio接口。



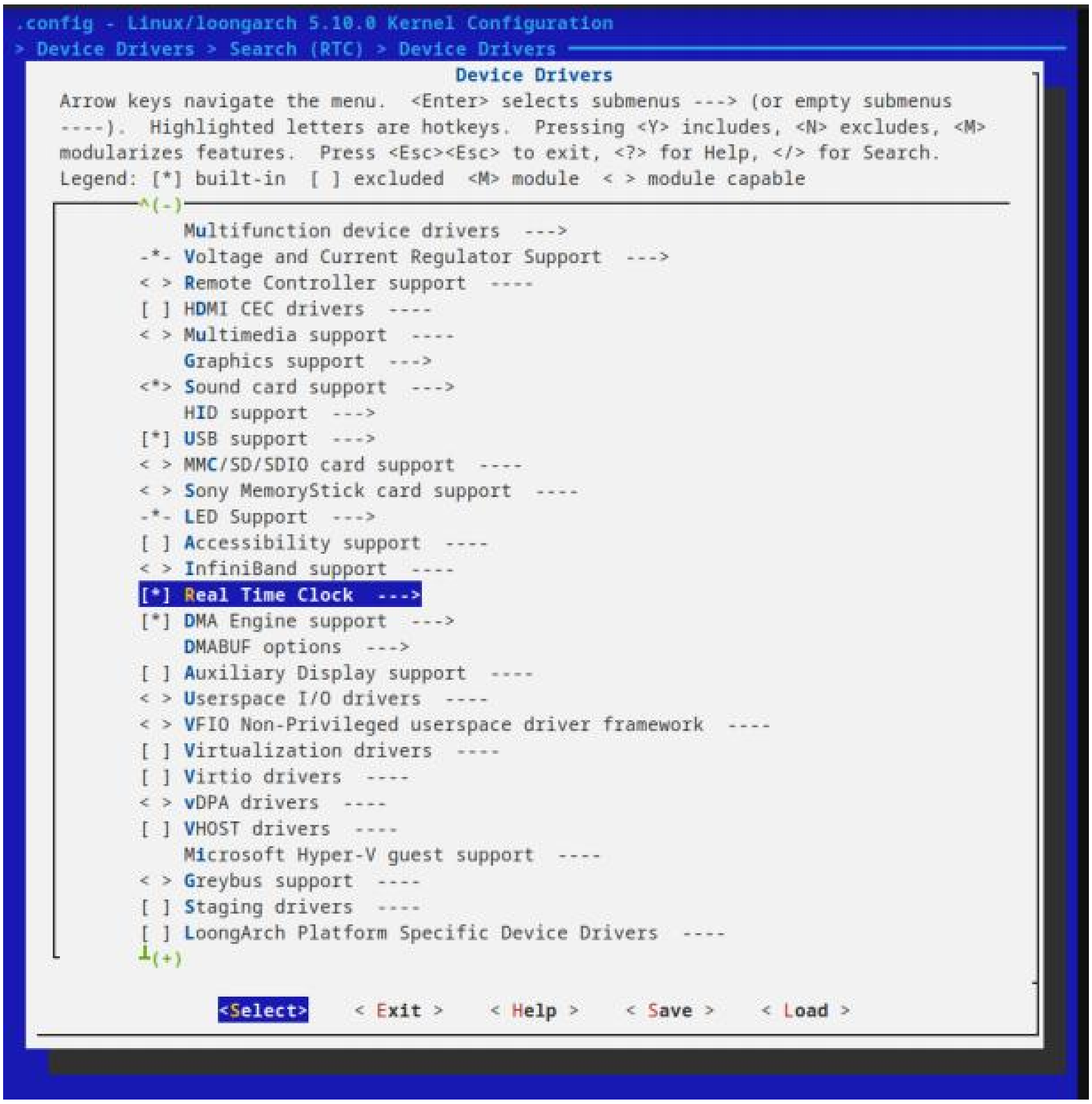
```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > GPIO Support ─────────────────────────────────────────
                              GPIO Support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- GPIO Support
        (512) Maximum number of GPIOs for fast path
        [ ]   Debug GPIO calls
        [*]   /sys/class/gpio/... (sysfs interface)
        [*]   Character device (/dev/gpiochipN) support
        [*]     Support GPIO ABI Version 1
              Memory mapped GPIO drivers  --->
              I2C GPIO expanders  --->
              MFD GPIO expanders  ----
              PCI GPIO expanders  --->
              SPI GPIO expanders  --->
              USB GPIO expanders  ----
        < >   GPIO Aggregator
        < >   GPIO Testing Driver

           <Select>    < Exit >    < Help >    < Save >    < Load >
```



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > GPIO Support > Memory mapped GPIO drivers
                          Memory mapped GPIO drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        < > GPIO driver for 74xx-ICs with MMIO access
        < > Altera GPIO
        < > AMD Promontory GPIO support
        < > Cadence GPIO support
        < > Synopsys DesignWare APB GPIO driver
        [ ] Faraday FTGPIO010 GPIO
        < > Generic memory-mapped GPIO controller support (MMIO platform device)
        < > Aeroflex Gaisler GRGPIO support
        < > Nintendo Wii (Hollywood) GPIO
        < > Xylon LogiCVC GPIO support
        [*] Loongson-2/3 GPIO support
        < > GPIO support for Fujitsu MB86S7x Platforms
        < > SAMA5D2 PIOBU GPIO support
        [ ] SiFive GPIO support
        < > GPIO based on SYSCON
        < > Xilinx GPIO support
        < > GPIO support for AMD Fusion Controller Hub (G-series SOCs)

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

10.2.2.RTC驱动

Device Driver 界面选中“Real Time Clock”选项，按空格选择为“*”，按回车进入

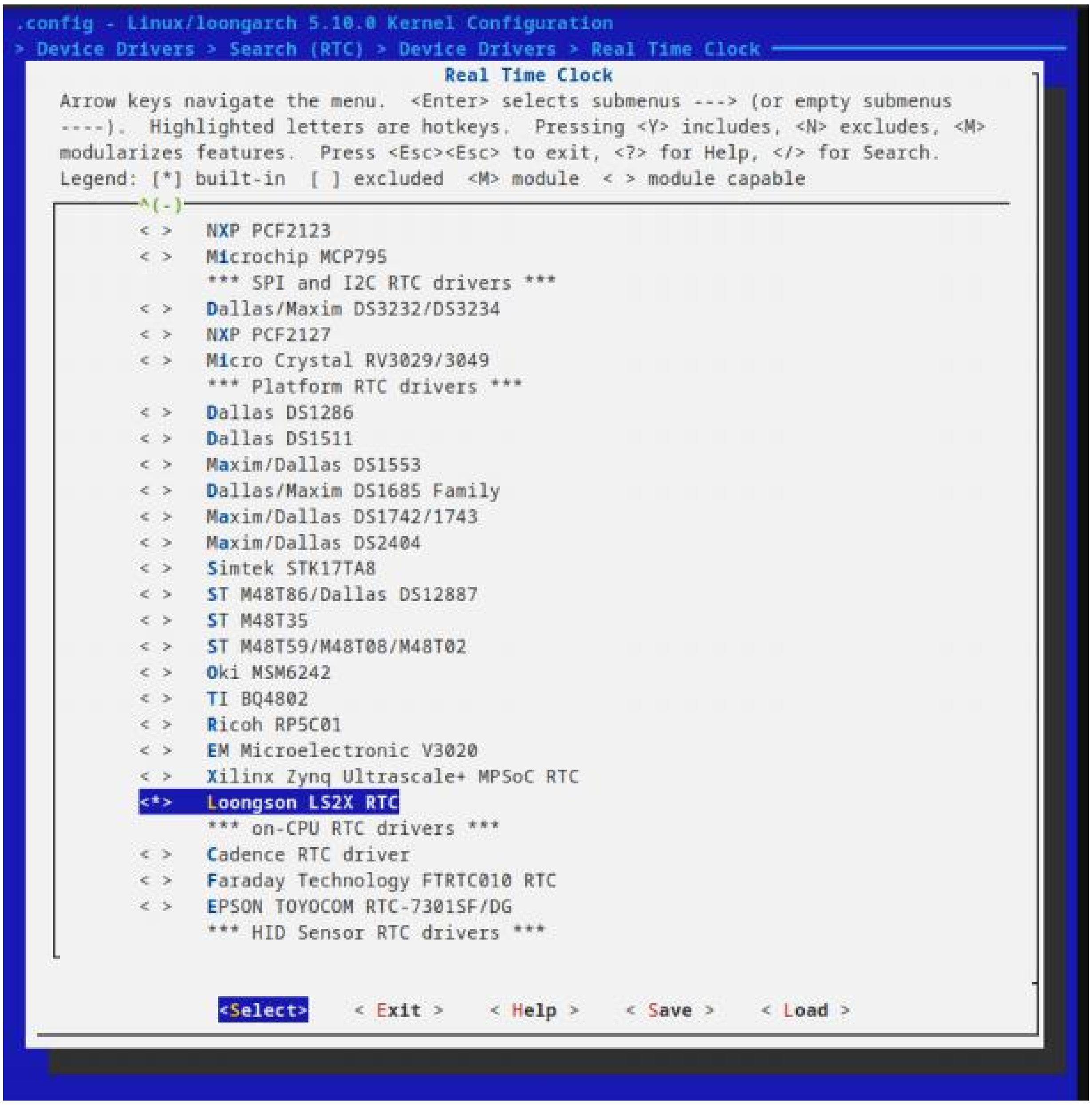

```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Search (RTC) > Device Drivers ─────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
 ┌───────^(-)──────────────────────────────────────────────────────────────────┐
 │          Multifunction device drivers  --->
 │      -*- Voltage and Current Regulator Support  --->
 │      < > Remote Controller support  ----
 │      [ ] HDMI CEC drivers  ----
 │      < > Multimedia support  ----
 │          Graphics support  --->
 │      <*> Sound card support  --->
 │          HID support  --->
 │      [*] USB support  --->
 │      < > MMC/SD/SDIO card support  ----
 │      < > Sony MemoryStick card support  ----
 │      -*- LED Support  --->
 │      [ ] Accessibility support  ----
 │      < > InfiniBand support  ----
 │      [*] Real Time Clock  --->
 │      [*] DMA Engine support  --->
 │          DMABUF options  --->
 │      [ ] Auxiliary Display support  ----
 │      < > Userspace I/O drivers  ----
 │      < > VFIO Non-Privileged userspace driver framework  ----
 │      [ ] Virtualization drivers  ----
 │      [ ] Virtio drivers  ----
 │      < > vDPA drivers  ----
 │      [ ] VHOST drivers  ----
 │          Microsoft Hyper-V guest support  ----
 │      < > Greybus support  ----
 │      [ ] Staging drivers  ----
 │      [ ] LoongArch Platform Specific Device Drivers  ----
 └───────┴(+)──────────────────────────────────────────────────────────────────┘
          <Select>    < Exit >    < Help >    < Save >    < Load >
```

以下选项的意义如下表：

选项	意义
Set system time from RTC on startup and resume	系统时间从RTC时间中获取
RTC used to set the system time	Set system time from RTC on startup and resume选中后，从哪个 RTC设备中读取RTC时间
Set the RTC time based on NTP synchronization	RTC时间从NTP同步服务中获取，即从网络上获取时间
RTC used to synchronize NTP adjustment	哪个RTC设备的时间从网络上获取
/sys/class/rtc/rtcN (sysfs)	提供sysfs接口给用户，可以设置，查询RTC的情况
/proc/driver/rtc (procfs for rtcN)	提供接口，可以使用cat /proc/driver/rtc查看rtc的情况

/dev/rtcN (character devices)	提供字符设备，可供用户对RTC的进行读写操作。

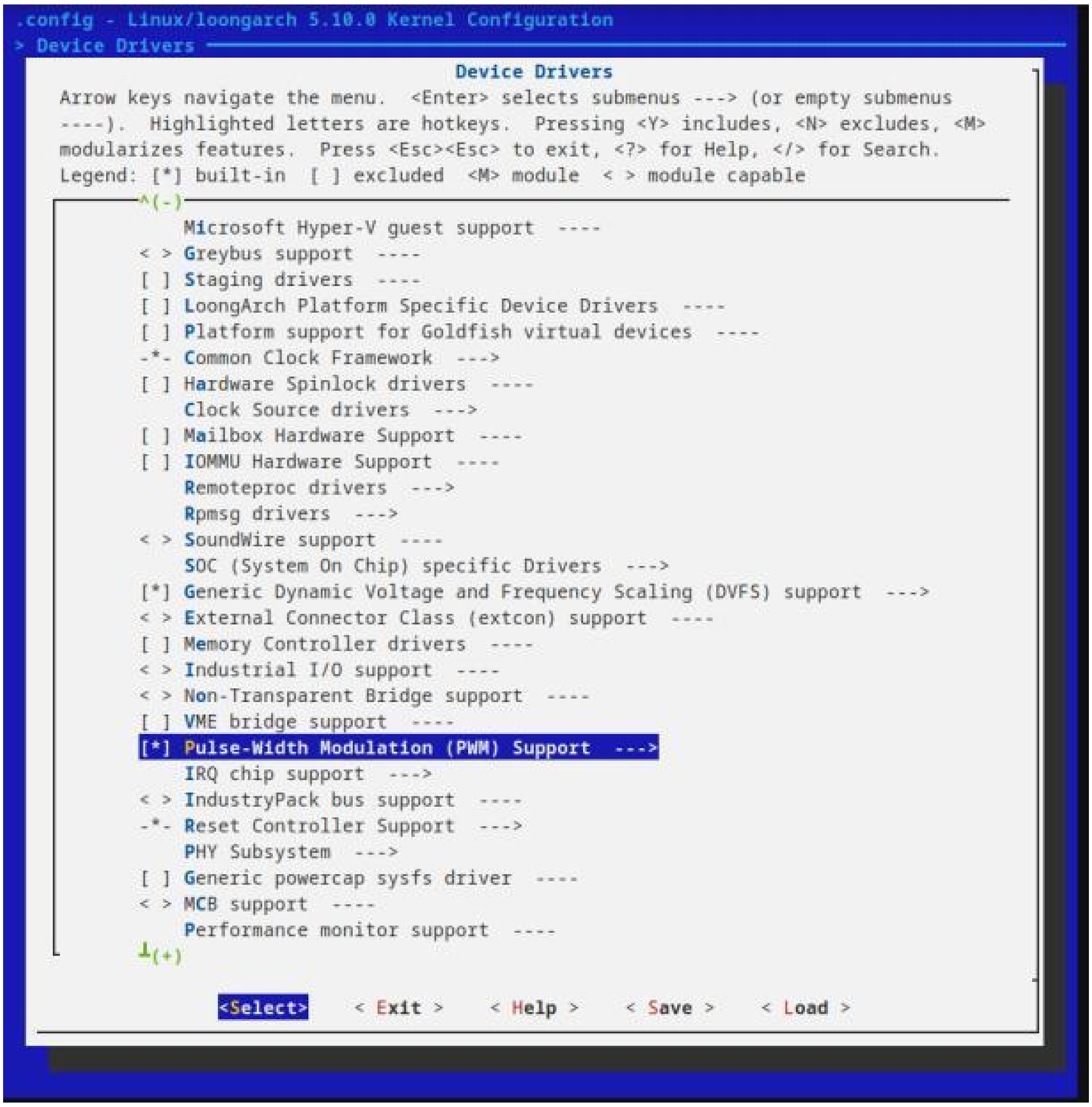
选中“loongson LS2X RTC”选项，按空格选择为“*”，即可使能龙芯2K1000上的RTC。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Search (RTC) > Device Drivers > Real Time Clock
                              Real Time Clock
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
         ^(-)
        < >   NXP PCF2123
        < >   Microchip MCP795
              *** SPI and I2C RTC drivers ***
        < >   Dallas/Maxim DS3232/DS3234
        < >   NXP PCF2127
        < >   Micro Crystal RV3029/3049
              *** Platform RTC drivers ***
        < >   Dallas DS1286
        < >   Dallas DS1511
        < >   Maxim/Dallas DS1553
        < >   Dallas/Maxim DS1685 Family
        < >   Maxim/Dallas DS1742/1743
        < >   Maxim/Dallas DS2404
        < >   Simtek STK17TA8
        < >   ST M48T86/Dallas DS12887
        < >   ST M48T35
        < >   ST M48T59/M48T08/M48T02
        < >   Oki MSM6242
        < >   TI BQ4802
        < >   Ricoh RP5C01
        < >   EM Microelectronic V3020
        < >   Xilinx Zynq Ultrascale+ MPSoC RTC
        <*>   Loongson LS2X RTC
              *** on-CPU RTC drivers ***
        < >   Cadence RTC driver
        < >   Faraday Technology FTRTC010 RTC
        < >   EPSON TOYOCOM RTC-7301SF/DG
              *** HID Sensor RTC drivers ***

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

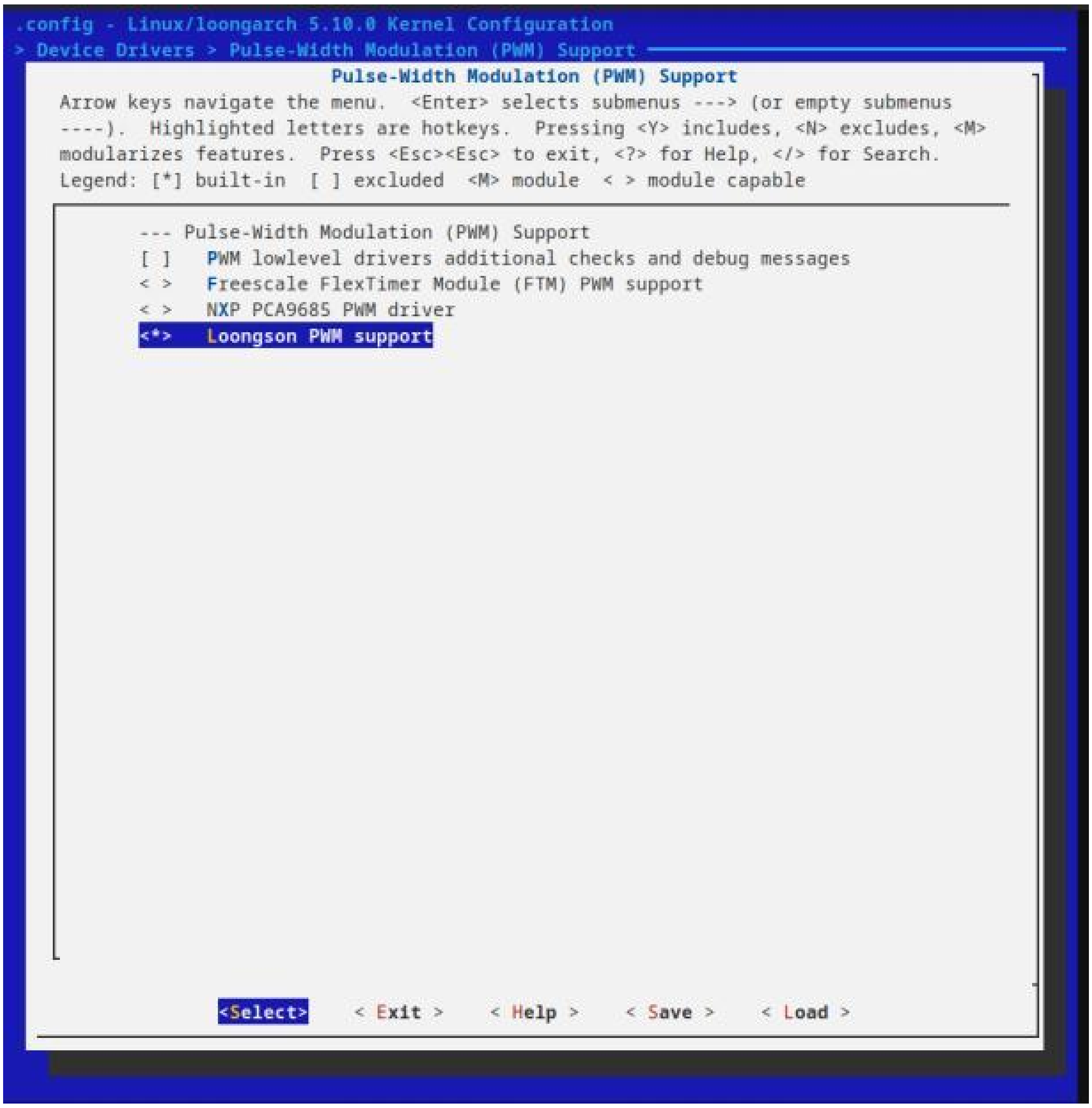
10.2.3.PWM驱动

Device Driver 界面选中“Pulse-Width Modulation (PWM) Support --->”选项，按空格选择为“*”，然后按回车进入。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers ─────────────────────────────────────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

      ^(-)
          Microsoft Hyper-V guest support  ----
      < > Greybus support  ----
      [ ] Staging drivers  ----
      [ ] LoongArch Platform Specific Device Drivers  ----
      [ ] Platform support for Goldfish virtual devices  ----
      -*- Common Clock Framework  --->
      [ ] Hardware Spinlock drivers  ----
          Clock Source drivers  --->
      [ ] Mailbox Hardware Support  ----
      [ ] IOMMU Hardware Support  ----
          Remoteproc drivers  --->
          Rpmsg drivers  --->
      < > SoundWire support  ----
          SOC (System On Chip) specific Drivers  --->
      [*] Generic Dynamic Voltage and Frequency Scaling (DVFS) support  --->
      < > External Connector Class (extcon) support  ----
      [ ] Memory Controller drivers  ----
      < > Industrial I/O support  ----
      < > Non-Transparent Bridge support  ----
      [ ] VME bridge support  ----
      [*] Pulse-Width Modulation (PWM) Support  --->
          IRQ chip support  --->
      < > IndustryPack bus support  ----
      -*- Reset Controller Support  --->
          PHY Subsystem  --->
      [ ] Generic powercap sysfs driver  ----
      < > MCB support  ----
          Performance monitor support  ----
      ┴(+)

         <Select>    < Exit >    < Help >    < Save >    < Load >
```

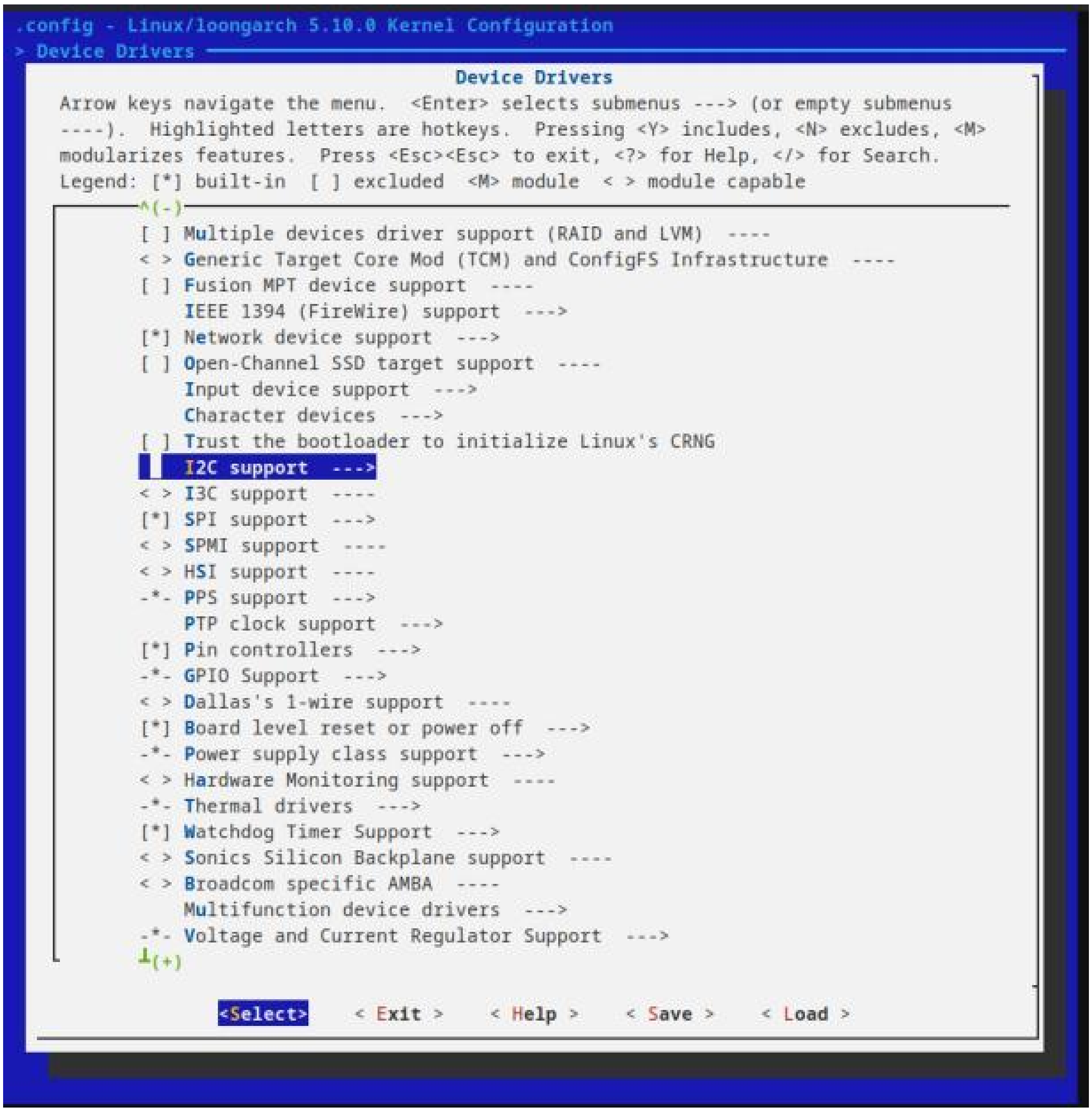
选中"Loongson PWM support"选项。按空格选为"*"。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Pulse-Width Modulation (PWM) Support ─────────────────────
                    Pulse-Width Modulation (PWM) Support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Pulse-Width Modulation (PWM) Support
        [ ]   PWM lowlevel drivers additional checks and debug messages
        < >   Freescale FlexTimer Module (FTM) PWM support
        < >   NXP PCA9685 PWM driver
        <*>   Loongson PWM support

            <Select>    < Exit >    < Help >    < Save >    < Load >
```

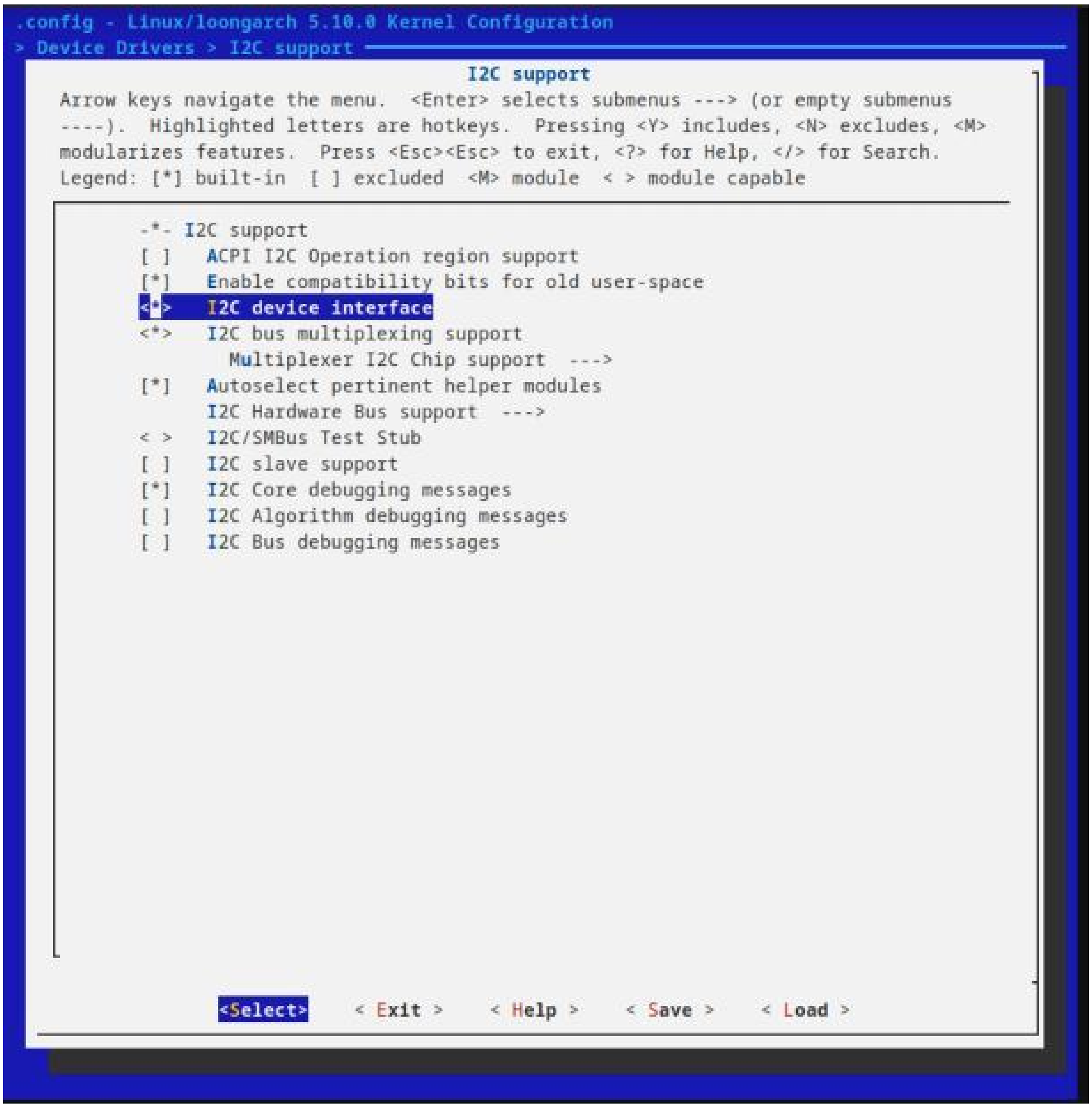
10.2.4. I2C驱动

Device Drivers 选中“I2C support”选项，按回车进入。



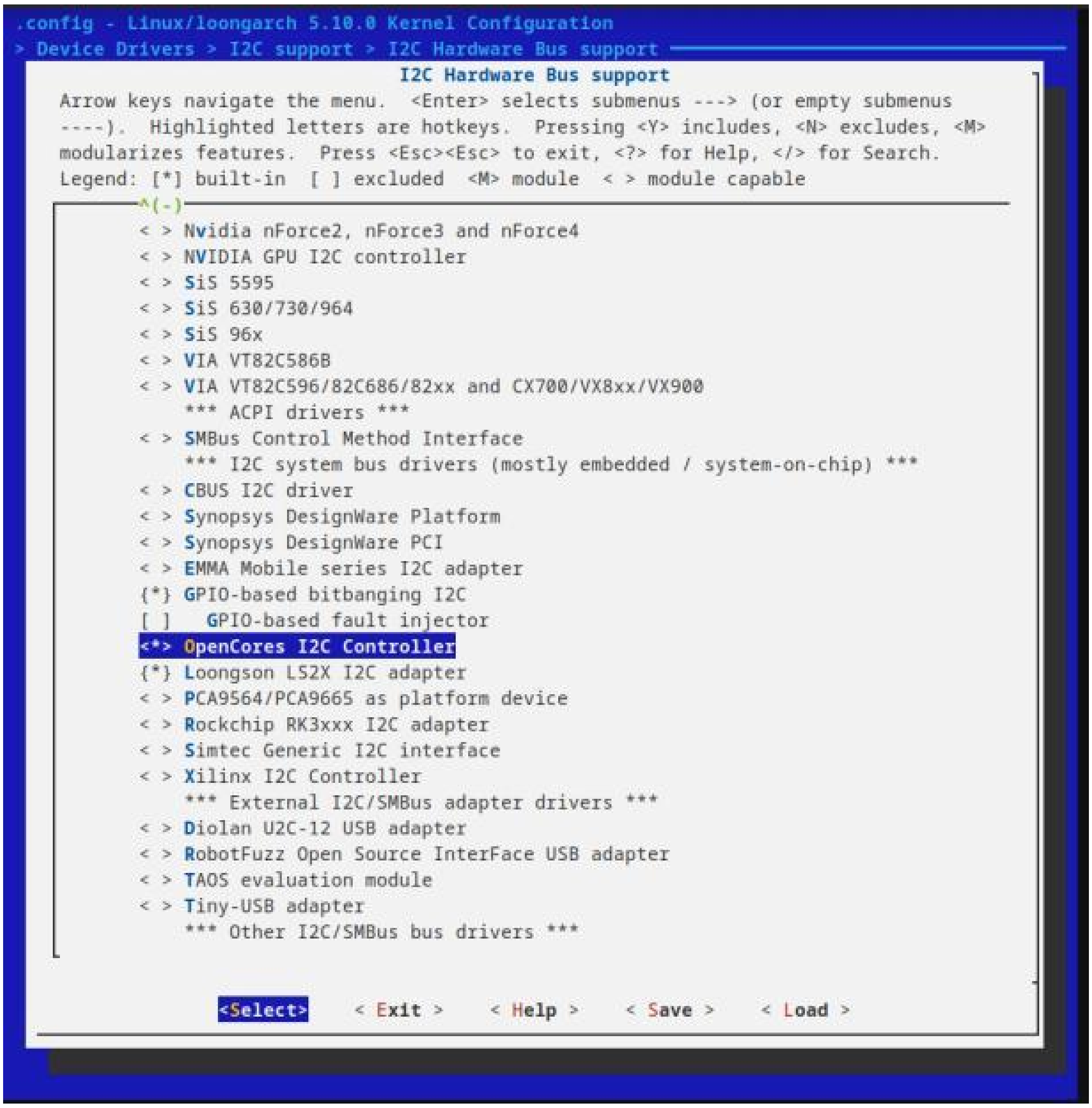
```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers ─────────────────────────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
        ^(-)
        [ ] Multiple devices driver support (RAID and LVM)  ----
        < > Generic Target Core Mod (TCM) and ConfigFS Infrastructure  ----
        [ ] Fusion MPT device support  ----
            IEEE 1394 (FireWire) support  --->
        [*] Network device support  --->
        [ ] Open-Channel SSD target support  ----
            Input device support  --->
            Character devices  --->
        [ ] Trust the bootloader to initialize Linux's CRNG
            I2C support  --->
        < > I3C support  ----
        [*] SPI support  --->
        < > SPMI support  ----
        < > HSI support  ----
        -*- PPS support  --->
            PTP clock support  --->
        [*] Pin controllers  --->
        -*- GPIO Support  --->
        < > Dallas's 1-wire support  ----
        [*] Board level reset or power off  --->
        -*- Power supply class support  --->
        < > Hardware Monitoring support  ----
        -*- Thermal drivers  --->
        [*] Watchdog Timer Support  --->
        < > Sonics Silicon Backplane support  ----
        < > Broadcom specific AMBA  ----
            Multifunction device drivers  --->
        -*- Voltage and Current Regulator Support  --->
        ┴(+)

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中“I2C device interface”，按空格选为“*”。选择此选项之后，系统启动后会生成/dev/i2c-x的设备。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > I2C support
                              I2C support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        -*- I2C support
        [ ]   ACPI I2C Operation region support
        [*]   Enable compatibility bits for old user-space
        <*>   I2C device interface
        <*>   I2C bus multiplexing support
                Multiplexer I2C Chip support  --->
        [*]   Autoselect pertinent helper modules
              I2C Hardware Bus support  --->
        < >   I2C/SMBus Test Stub
        [ ]   I2C slave support
        [*]   I2C Core debugging messages
        [ ]   I2C Algorithm debugging messages
        [ ]   I2C Bus debugging messages

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

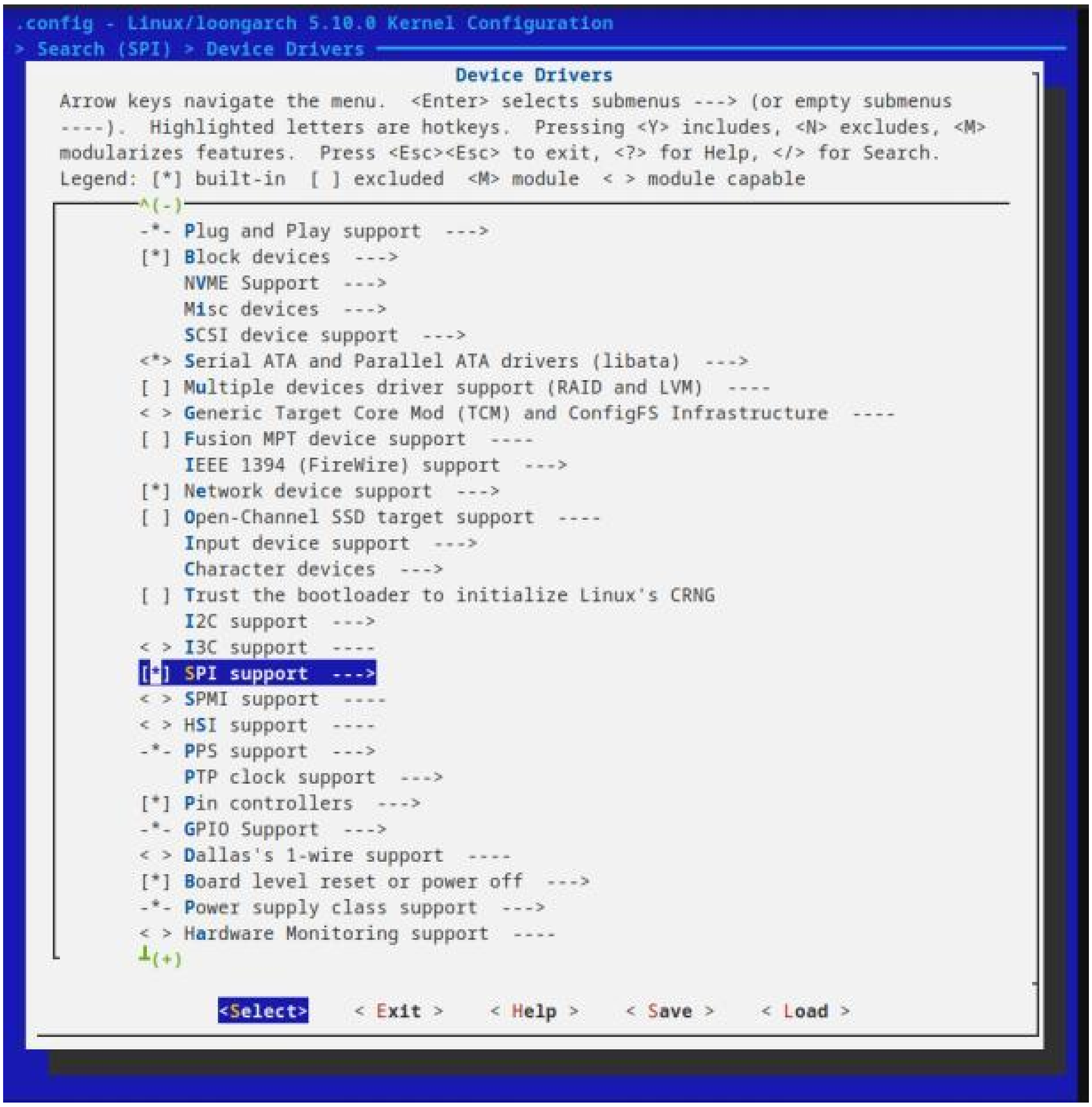
进入 I2C Hardware Bus support， 选择 OpenCores I2C Controller 与 Loongson LS2X I2C adapter ![img](res/images/十一-i2c-hardware-bus.png)



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > I2C support > I2C Hardware Bus support ─────────────────
                         I2C Hardware Bus support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
        ^(-)
        < > Nvidia nForce2, nForce3 and nForce4
        < > NVIDIA GPU I2C controller
        < > SiS 5595
        < > SiS 630/730/964
        < > SiS 96x
        < > VIA VT82C586B
        < > VIA VT82C596/82C686/82xx and CX700/VX8xx/VX900
            *** ACPI drivers ***
        < > SMBus Control Method Interface
            *** I2C system bus drivers (mostly embedded / system-on-chip) ***
        < > CBUS I2C driver
        < > Synopsys DesignWare Platform
        < > Synopsys DesignWare PCI
        < > EMMA Mobile series I2C adapter
        {*} GPIO-based bitbanging I2C
        [ ]   GPIO-based fault injector
        <*> OpenCores I2C Controller
        {*} Loongson LS2X I2C adapter
        < > PCA9564/PCA9665 as platform device
        < > Rockchip RK3xxx I2C adapter
        < > Simtec Generic I2C interface
        < > Xilinx I2C Controller
            *** External I2C/SMBus adapter drivers ***
        < > Diolan U2C-12 USB adapter
        < > RobotFuzz Open Source InterFace USB adapter
        < > TAOS evaluation module
        < > Tiny-USB adapter
            *** Other I2C/SMBus bus drivers ***

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

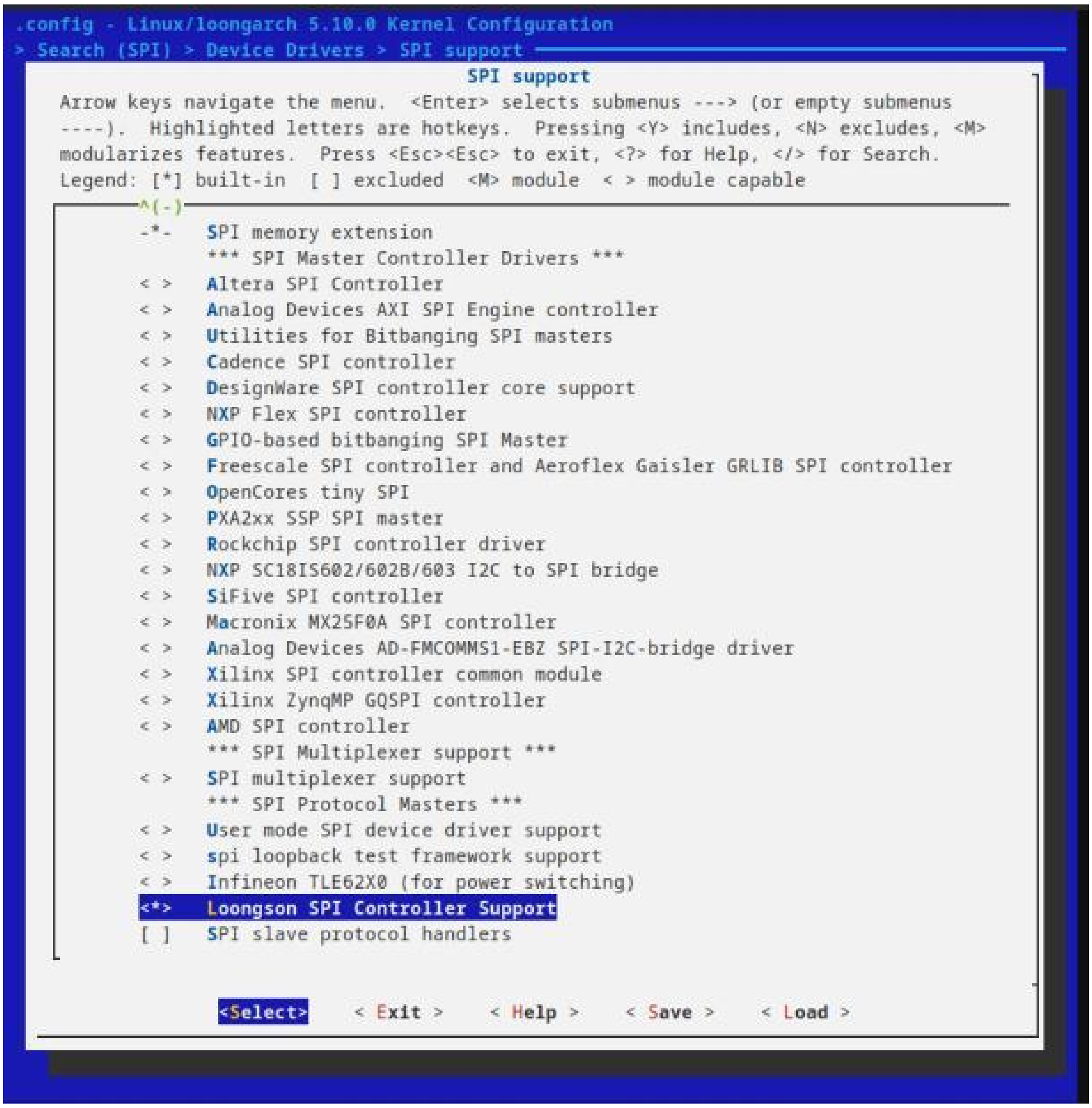
10.2.5.SPI驱动

Device Driver 选中"SPI support"选项，按空格选为"*"，按回车进入。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Search (SPI) > Device Drivers ─────────────────────────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        ^(-)
        -*- Plug and Play support  --->
        [*] Block devices  --->
            NVME Support  --->
            Misc devices  --->
            SCSI device support  --->
        <*> Serial ATA and Parallel ATA drivers (libata)  --->
        [ ] Multiple devices driver support (RAID and LVM)  ----
        < > Generic Target Core Mod (TCM) and ConfigFS Infrastructure  ----
        [ ] Fusion MPT device support  ----
            IEEE 1394 (FireWire) support  --->
        [*] Network device support  --->
        [ ] Open-Channel SSD target support  ----
            Input device support  --->
            Character devices  --->
        [ ] Trust the bootloader to initialize Linux's CRNG
            I2C support  --->
        < > I3C support  ----
        [*] SPI support  --->
        < > SPMI support  ----
        < > HSI support  ----
        -*- PPS support  --->
            PTP clock support  --->
        [*] Pin controllers  --->
        -*- GPIO Support  --->
        < > Dallas's 1-wire support  ----
        [*] Board level reset or power off  --->
        -*- Power supply class support  --->
        < > Hardware Monitoring support  ----
        v(+)

              <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中“Loongson SPI controller Support”选项，按空格选为“*”，启用龙芯2K1000的自带的SPI控制器。



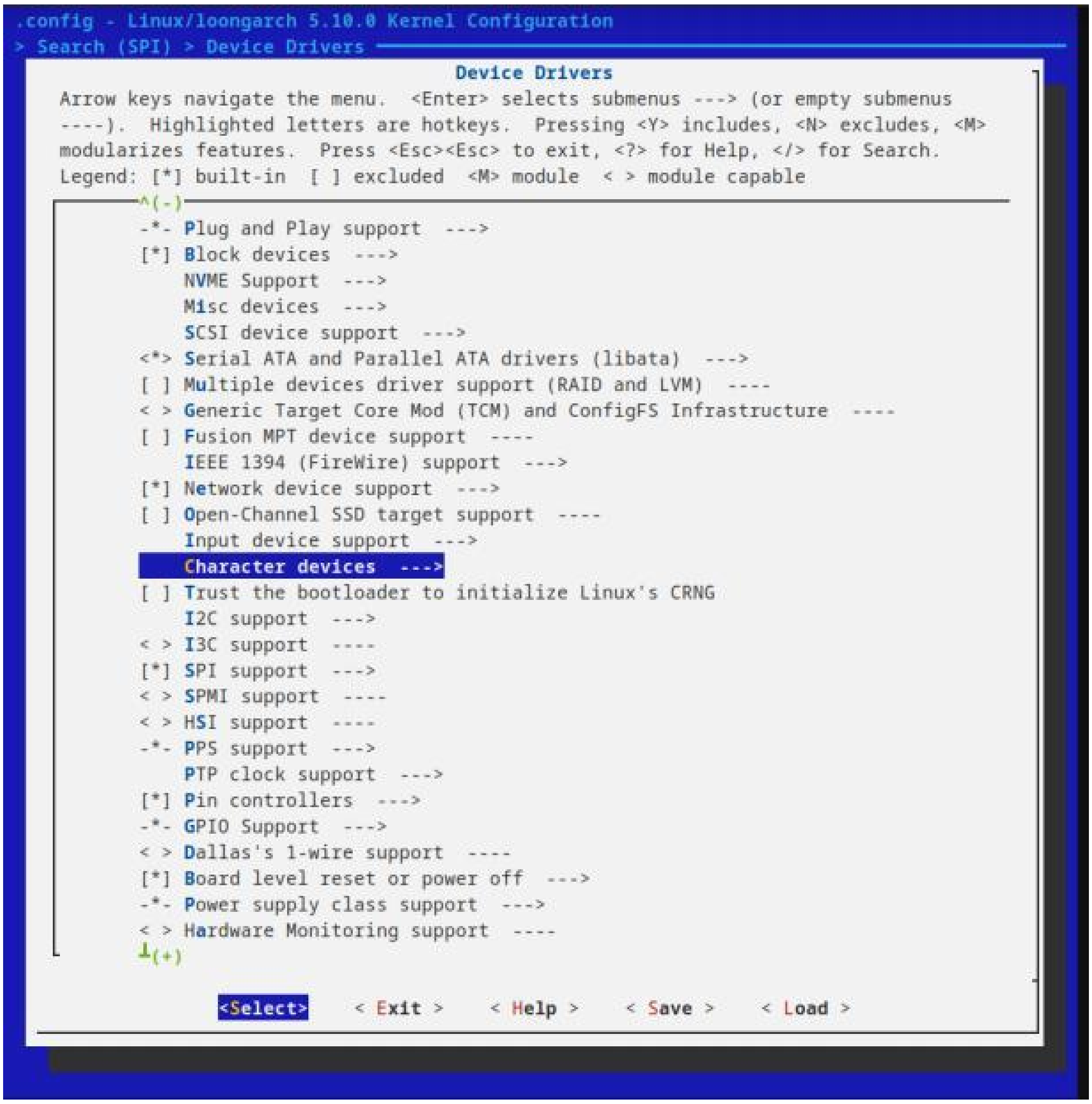
```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Search (SPI) > Device Drivers > SPI support ─────────────────────────────
                              SPI support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
          ^(-)
         -*-    SPI memory extension
                *** SPI Master Controller Drivers ***
         < >    Altera SPI Controller
         < >    Analog Devices AXI SPI Engine controller
         < >    Utilities for Bitbanging SPI masters
         < >    Cadence SPI controller
         < >    DesignWare SPI controller core support
         < >    NXP Flex SPI controller
         < >    GPIO-based bitbanging SPI Master
         < >    Freescale SPI controller and Aeroflex Gaisler GRLIB SPI controller
         < >    OpenCores tiny SPI
         < >    PXA2xx SSP SPI master
         < >    Rockchip SPI controller driver
         < >    NXP SC18IS602/602B/603 I2C to SPI bridge
         < >    SiFive SPI controller
         < >    Macronix MX25F0A SPI controller
         < >    Analog Devices AD-FMCOMMS1-EBZ SPI-I2C-bridge driver
         < >    Xilinx SPI controller common module
         < >    Xilinx ZynqMP GQSPI controller
         < >    AMD SPI controller
                *** SPI Multiplexer support ***
         < >    SPI multiplexer support
                *** SPI Protocol Masters ***
         < >    User mode SPI device driver support
         < >    spi loopback test framework support
         < >    Infineon TLE62X0 (for power switching)
         <*>    Loongson SPI Controller Support
         [ ]    SPI slave protocol handlers

              <Select>    < Exit >    < Help >    < Save >    < Load >
```

对于SPI接口，在金龙板上SPI0总线的片选0是用于SPI-FLASH的片选。然后SPI-FLASH会当作MTD，从而在/dev中是没有 spidev0.0这个设备。对应的是mtd4、 mtdblock4。其余的SPI片选1、2和3在扩展接口板接口中。请见于第三章的第18　　点—扩展板接口。

由于SPI-FLASH里面存储的是uboot，那么不建议对SPI-FLASH进行写操作，避免对uboot造成破坏，从而不能启动系统。 如果需要进行写操作，需要注意的是擦写大小为4KB。

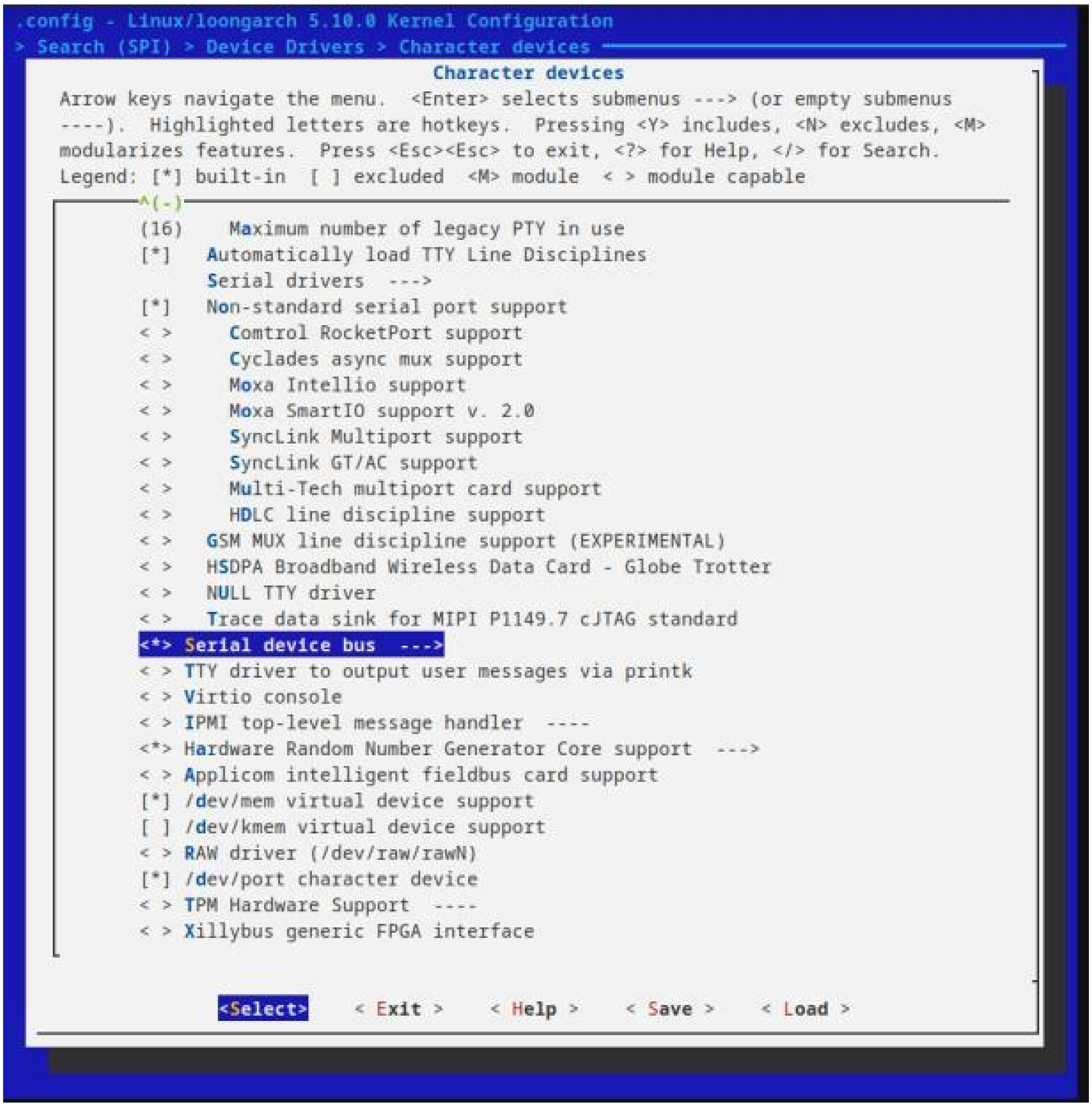
10.2.6. UART驱动

Device Driver 选中"Character devices"选项，按回车进入。

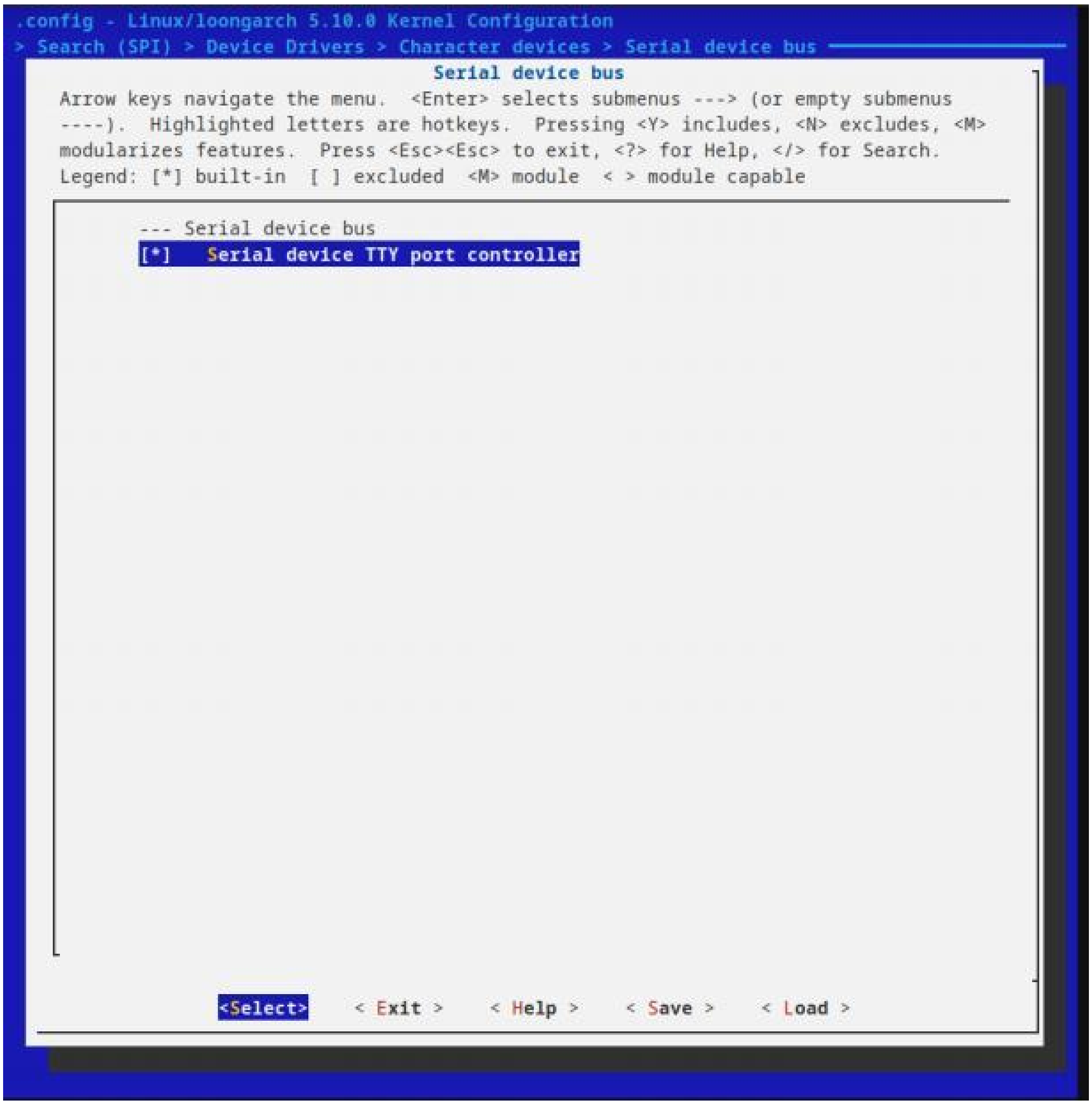


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Search (SPI) > Device Drivers ───────────────────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        ^(-)
        -*- Plug and Play support  --->
        [*] Block devices  --->
            NVME Support  --->
            Misc devices  --->
            SCSI device support  --->
        <*> Serial ATA and Parallel ATA drivers (libata)  --->
        [ ] Multiple devices driver support (RAID and LVM)  ----
        < > Generic Target Core Mod (TCM) and ConfigFS Infrastructure  ----
        [ ] Fusion MPT device support  ----
            IEEE 1394 (FireWire) support  --->
        [*] Network device support  --->
        [ ] Open-Channel SSD target support  ----
            Input device support  --->
            Character devices  --->
        [ ] Trust the bootloader to initialize Linux's CRNG
            I2C support  --->
        < > I3C support  ----
        [*] SPI support  --->
        < > SPMI support  ----
        < > HSI support  ----
        -*- PPS support  --->
            PTP clock support  --->
        [*] Pin controllers  --->
        -*- GPIO Support  --->
        < > Dallas's 1-wire support  ----
        [*] Board level reset or power off  --->
        -*- Power supply class support  --->
        < > Hardware Monitoring support  ----
        ┴(+)

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中"Serial device bus"选项，按回车进入，选择"Serial device TTY port controller"



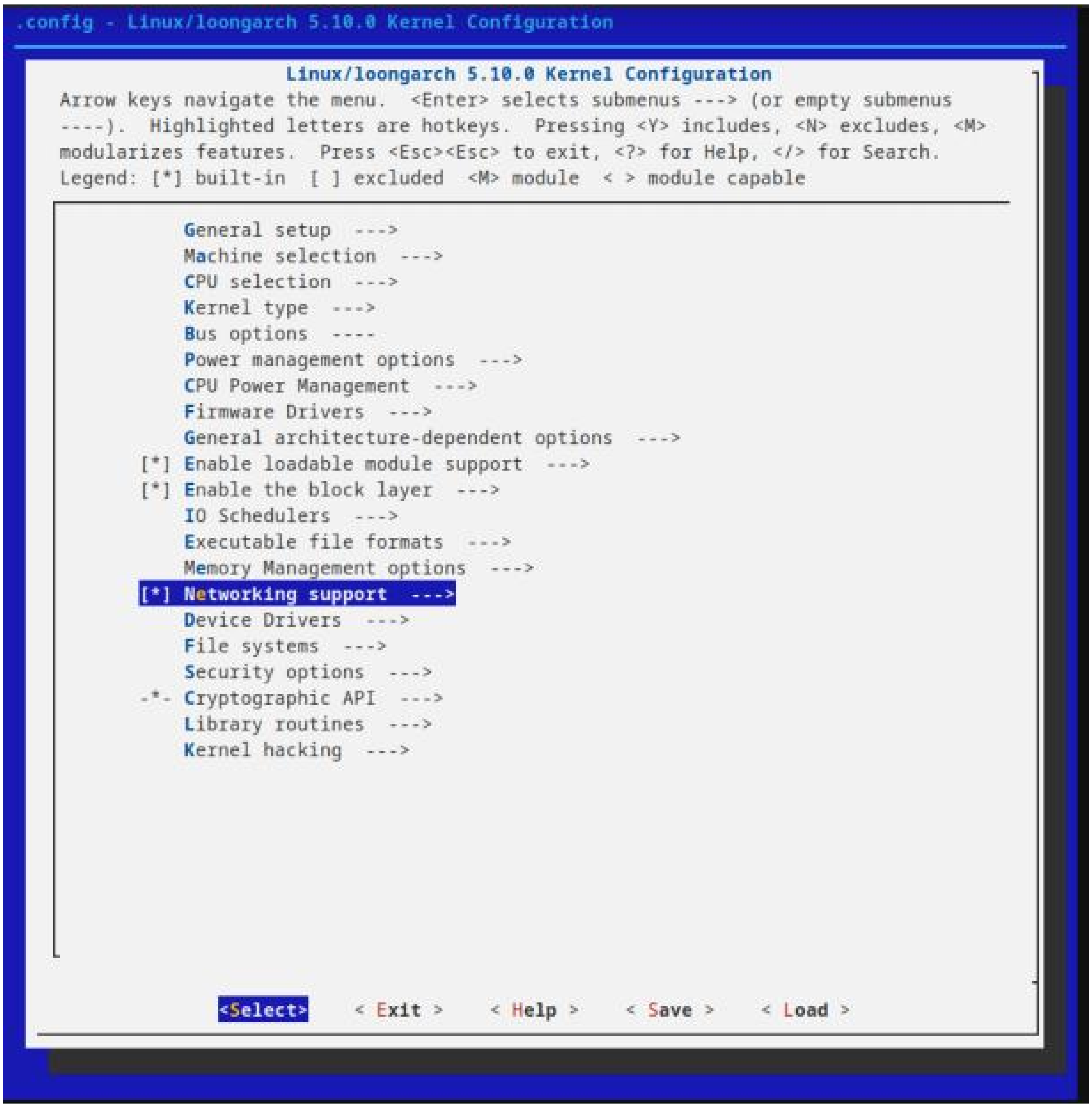
```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Search (SPI) > Device Drivers > Character devices ──────────────────────────
                              Character devices
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        ^(-)
        (16)    Maximum number of legacy PTY in use
        [*]   Automatically load TTY Line Disciplines
              Serial drivers  --->
        [*]   Non-standard serial port support
        < >     Comtrol RocketPort support
        < >     Cyclades async mux support
        < >     Moxa Intellio support
        < >     Moxa SmartIO support v. 2.0
        < >     SyncLink Multiport support
        < >     SyncLink GT/AC support
        < >     Multi-Tech multiport card support
        < >     HDLC line discipline support
        < >   GSM MUX line discipline support (EXPERIMENTAL)
        < >   HSDPA Broadband Wireless Data Card - Globe Trotter
        < >   NULL TTY driver
        < >   Trace data sink for MIPI P1149.7 cJTAG standard
        <*> Serial device bus  --->
        < > TTY driver to output user messages via printk
        < > Virtio console
        < > IPMI top-level message handler  ----
        <*> Hardware Random Number Generator Core support  --->
        < > Applicom intelligent fieldbus card support
        [*] /dev/mem virtual device support
        [ ] /dev/kmem virtual device support
        < > RAW driver (/dev/raw/rawN)
        [*] /dev/port character device
        < > TPM Hardware Support  ----
        < > Xillybus generic FPGA interface

             <Select>    < Exit >    < Help >    < Save >    < Load >
```



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Search (SPI) > Device Drivers > Character devices > Serial device bus
                              Serial device bus
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Serial device bus
        [*]   Serial device TTY port controller

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

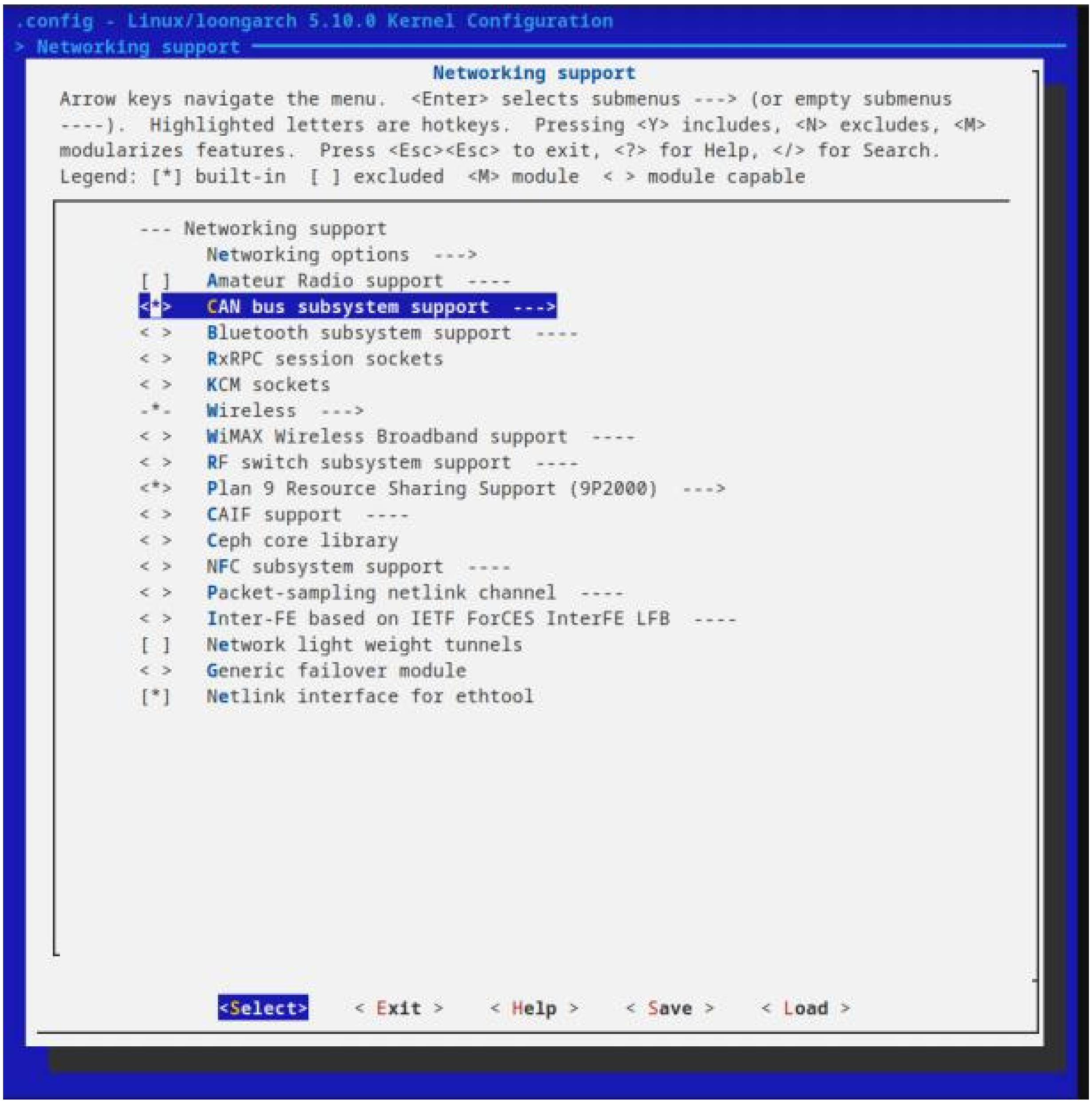
10.2.7.CAN驱动

在主菜单页面中选中"Networking support"选项，按回车进入。



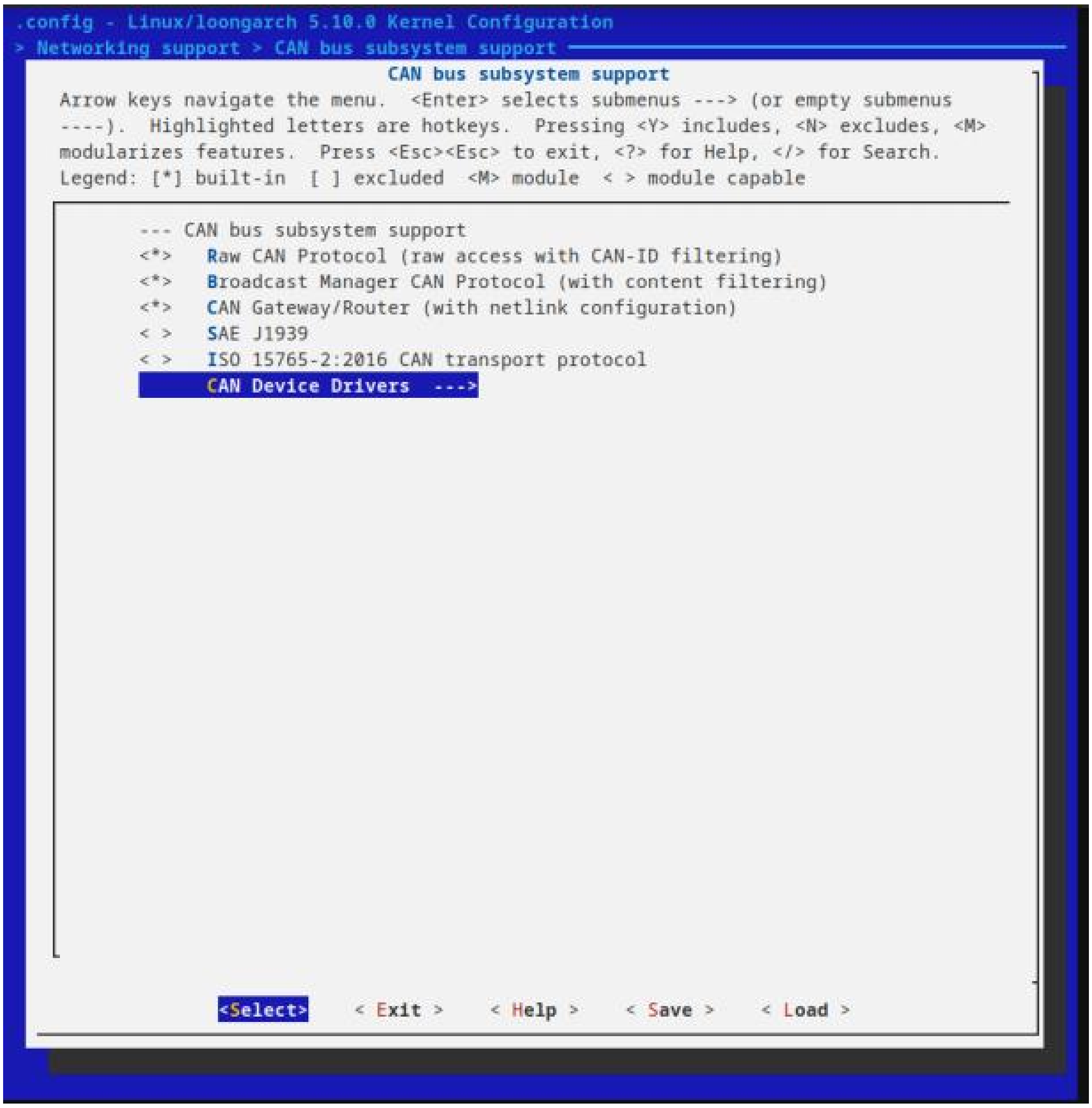
```
.config - Linux/loongarch 5.10.0 Kernel Configuration
                    Linux/loongarch 5.10.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

              General setup  --->
              Machine selection  --->
              CPU selection  --->
              Kernel type  --->
              Bus options  ----
              Power management options  --->
              CPU Power Management  --->
              Firmware Drivers  --->
              General architecture-dependent options  --->
          [*] Enable loadable module support  --->
          [*] Enable the block layer  --->
              IO Schedulers  --->
              Executable file formats  --->
              Memory Management options  --->
          [*] Networking support  --->
              Device Drivers  --->
              File systems  --->
              Security options  --->
          -*- Cryptographic API  --->
              Library routines  --->
              Kernel hacking  --->

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中“CAN bus subsystem support”，选为“*”，然后回车进入



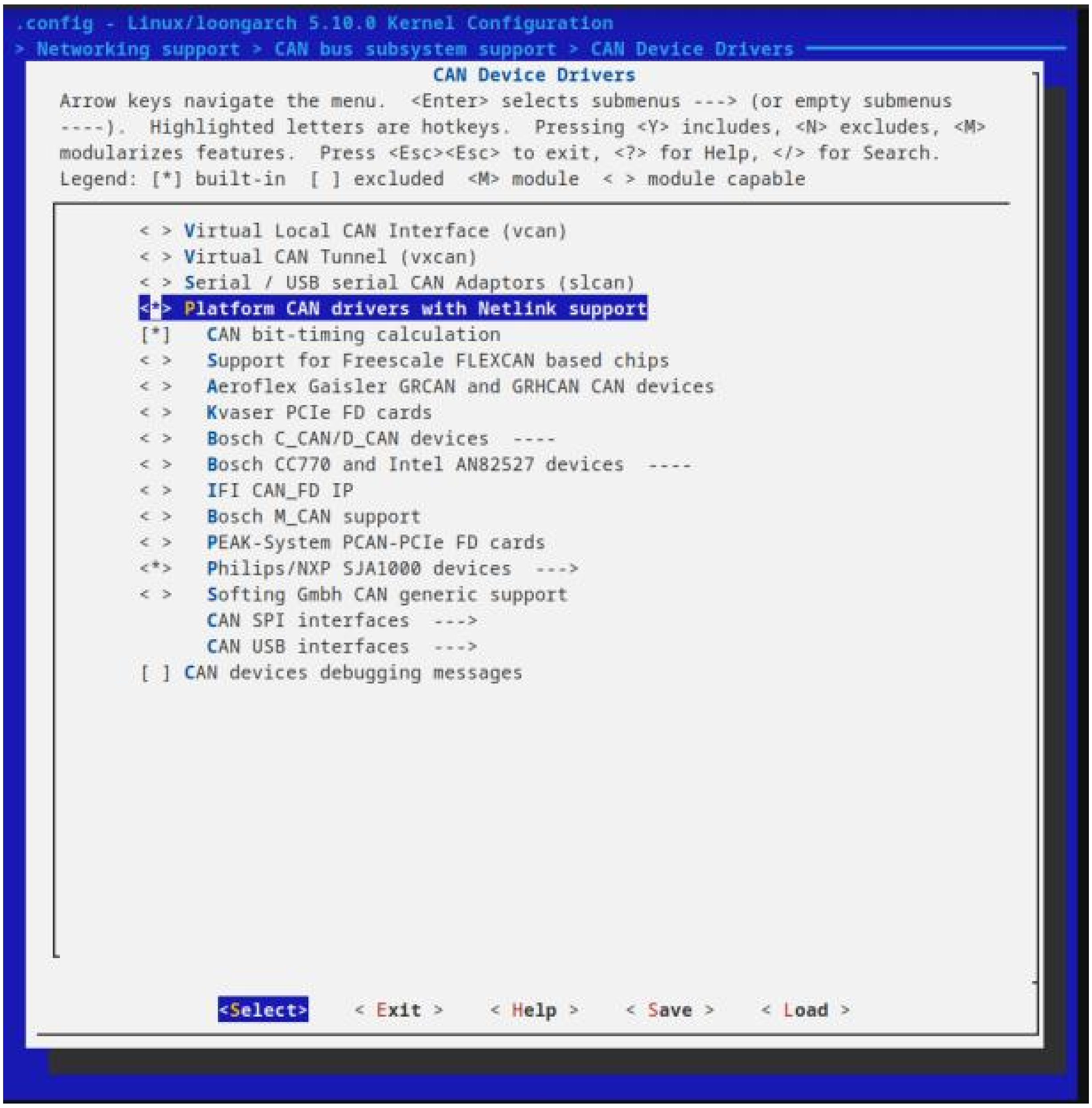
```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Networking support
                          Networking support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Networking support
              Networking options  --->
        [ ]   Amateur Radio support  ----
        <*>   CAN bus subsystem support  --->
        < >   Bluetooth subsystem support  ----
        < >   RxRPC session sockets
        < >   KCM sockets
        -*-   Wireless  --->
        < >   WiMAX Wireless Broadband support  ----
        < >   RF switch subsystem support  ----
        <*>   Plan 9 Resource Sharing Support (9P2000)  --->
        < >   CAIF support  ----
        < >   Ceph core library
        < >   NFC subsystem support  ----
        < >   Packet-sampling netlink channel  ----
        < >   Inter-FE based on IETF ForCES InterFE LFB  ----
        [ ]   Network light weight tunnels
        < >   Generic failover module
        [*]   Netlink interface for ethtool

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

按下图配置，然后选中“*CAN Device Drivers*”选项，然后按回车进入。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Networking support > CAN bus subsystem support ─────────────────────
                          CAN bus subsystem support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- CAN bus subsystem support
        <*>   Raw CAN Protocol (raw access with CAN-ID filtering)
        <*>   Broadcast Manager CAN Protocol (with content filtering)
        <*>   CAN Gateway/Router (with netlink configuration)
        < >   SAE J1939
        < >   ISO 15765-2:2016 CAN transport protocol
              CAN Device Drivers  --->

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

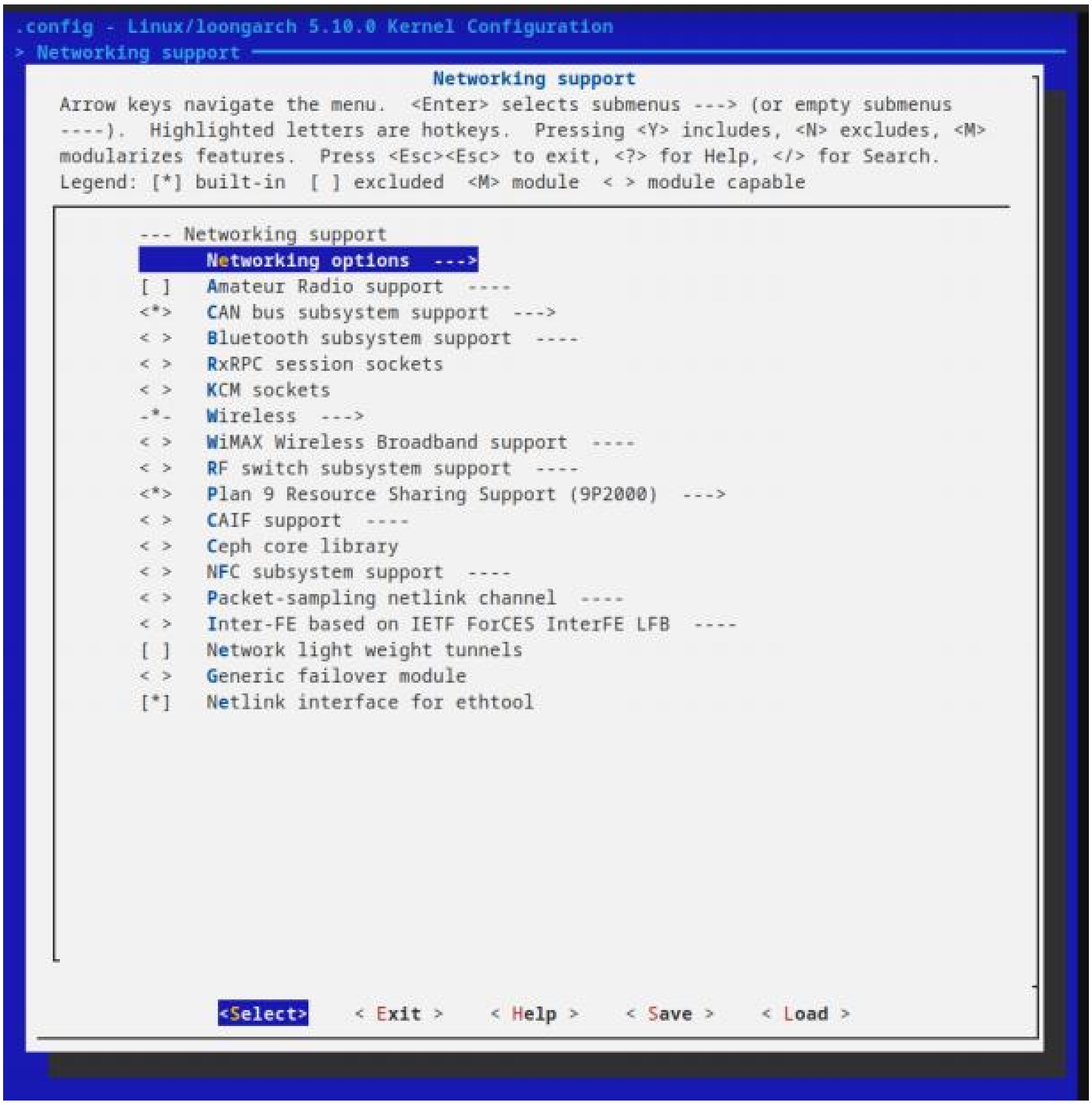
按下图配置。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Networking support > CAN bus subsystem support > CAN Device Drivers ──────────
                              CAN Device Drivers
    Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
    ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
    modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
    Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

          < > Virtual Local CAN Interface (vcan)
          < > Virtual CAN Tunnel (vxcan)
          < > Serial / USB serial CAN Adaptors (slcan)
          <*> Platform CAN drivers with Netlink support
          [*]   CAN bit-timing calculation
          < >   Support for Freescale FLEXCAN based chips
          < >   Aeroflex Gaisler GRCAN and GRHCAN CAN devices
          < >   Kvaser PCIe FD cards
          < >   Bosch C_CAN/D_CAN devices  ----
          < >   Bosch CC770 and Intel AN82527 devices  ----
          < >   IFI CAN_FD IP
          < >   Bosch M_CAN support
          < >   PEAK-System PCAN-PCIe FD cards
          <*>   Philips/NXP SJA1000 devices  --->
          < >   Softing Gmbh CAN generic support
                CAN SPI interfaces  --->
                CAN USB interfaces  --->
          [ ] CAN devices debugging messages

             <Select>    < Exit >    < Help >    < Save >    < Load >
```

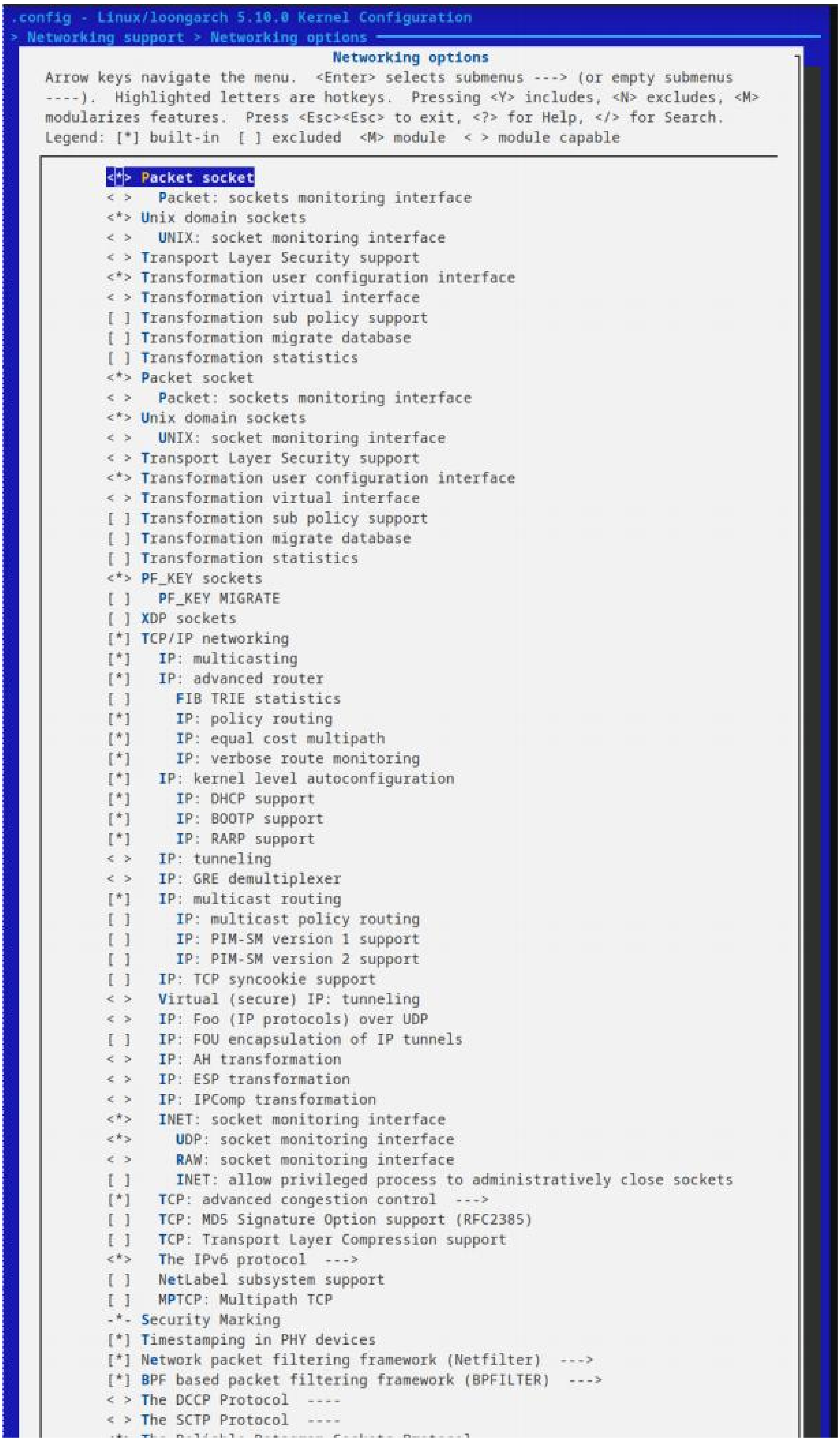
10.2.8.GMAC驱动

Networking support 选中“Networking options”，按回车进入。

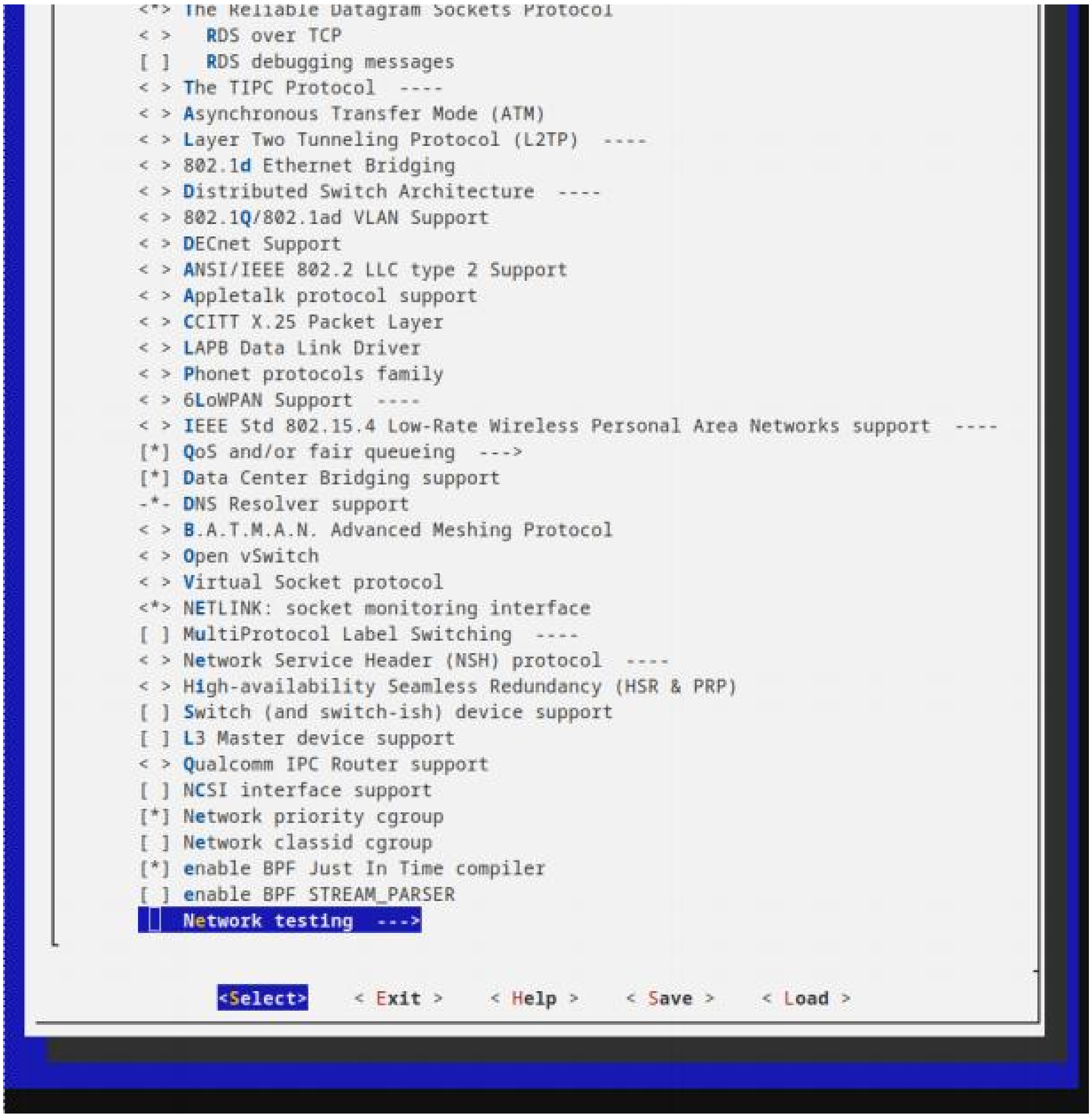


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Networking support ─────────────────────────────────────────────────────
                            Networking support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Networking support
              Networking options  --->
        [ ]   Amateur Radio support  ----
        <*>   CAN bus subsystem support  --->
        < >   Bluetooth subsystem support  ----
        < >   RxRPC session sockets
        < >   KCM sockets
        -*-   Wireless  --->
        < >   WiMAX Wireless Broadband support  ----
        < >   RF switch subsystem support  ----
        <*>   Plan 9 Resource Sharing Support (9P2000)  --->
        < >   CAIF support  ----
        < >   Ceph core library
        < >   NFC subsystem support  ----
        < >   Packet-sampling netlink channel  ----
        < >   Inter-FE based on IETF ForCES InterFE LFB  ----
        [ ]   Network light weight tunnels
        < >   Generic failover module
        [*]   Netlink interface for ethtool

             <Select>    < Exit >    < Help >    < Save >    < Load >
```

配置协议支持，按照下图中选择了“*”的选项来配置。

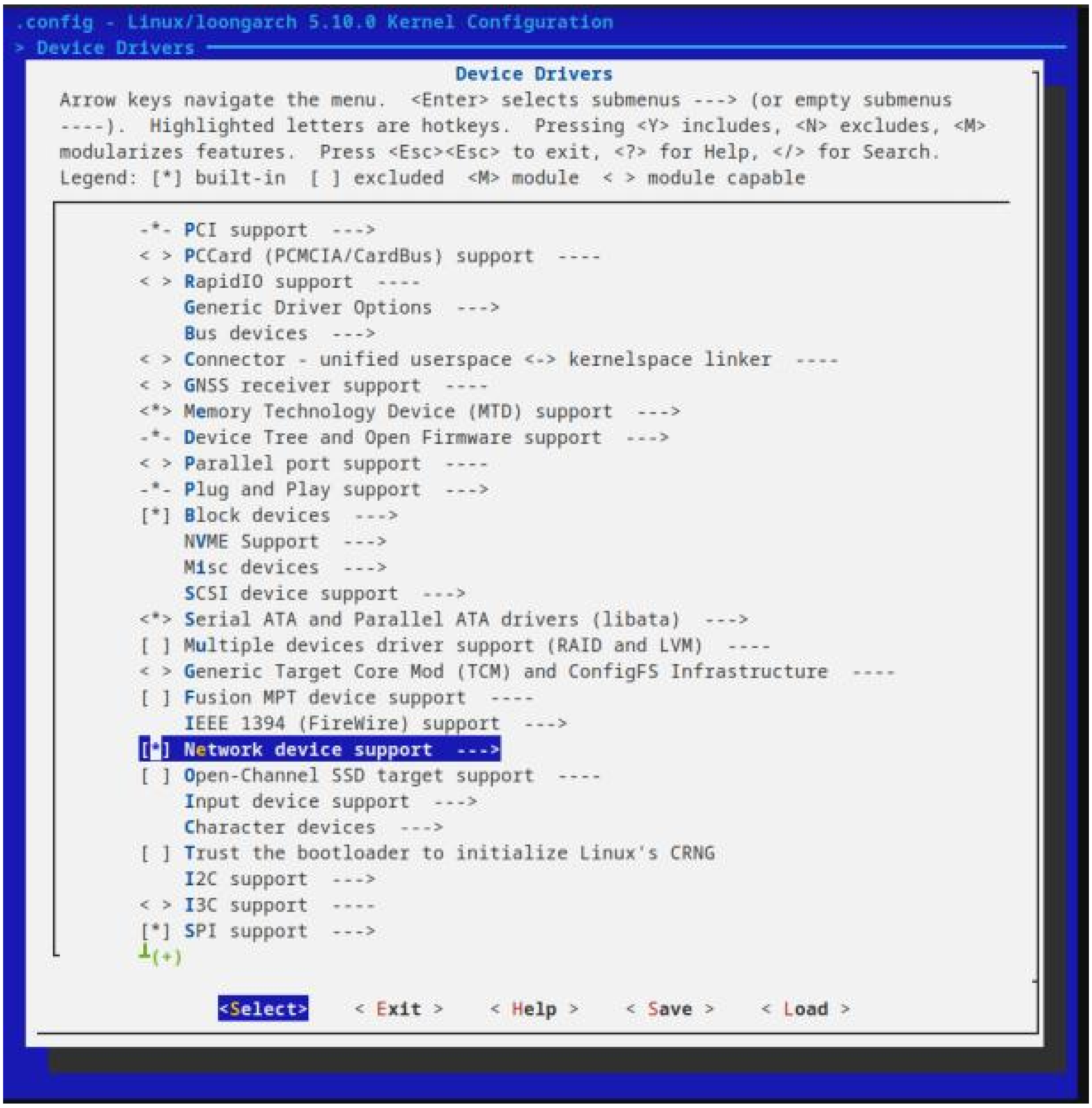


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Networking support > Networking options ─────────────────────────────────────
                              Networking options
   Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
   ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
   modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
   Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

          <*> Packet socket
          < >   Packet: sockets monitoring interface
          <*> Unix domain sockets
          < >   UNIX: socket monitoring interface
          < > Transport Layer Security support
          <*> Transformation user configuration interface
          < > Transformation virtual interface
          [ ] Transformation sub policy support
          [ ] Transformation migrate database
          [ ] Transformation statistics
          <*> Packet socket
          < >   Packet: sockets monitoring interface
          <*> Unix domain sockets
          < >   UNIX: socket monitoring interface
          < > Transport Layer Security support
          <*> Transformation user configuration interface
          < > Transformation virtual interface
          [ ] Transformation sub policy support
          [ ] Transformation migrate database
          [ ] Transformation statistics
          <*> PF_KEY sockets
          [ ]   PF_KEY MIGRATE
          [ ] XDP sockets
          [*] TCP/IP networking
          [*]   IP: multicasting
          [*]   IP: advanced router
          [ ]     FIB TRIE statistics
          [*]     IP: policy routing
          [*]     IP: equal cost multipath
          [*]     IP: verbose route monitoring
          [*]   IP: kernel level autoconfiguration
          [*]     IP: DHCP support
          [*]     IP: BOOTP support
          [*]     IP: RARP support
          < >   IP: tunneling
          < >   IP: GRE demultiplexer
          [*]   IP: multicast routing
          [ ]     IP: multicast policy routing
          [ ]     IP: PIM-SM version 1 support
          [ ]     IP: PIM-SM version 2 support
          [ ]   IP: TCP syncookie support
          < >   Virtual (secure) IP: tunneling
          < >   IP: Foo (IP protocols) over UDP
          [ ]   IP: FOU encapsulation of IP tunnels
          < >   IP: AH transformation
          < >   IP: ESP transformation
          < >   IP: IPComp transformation
          <*>   INET: socket monitoring interface
          <*>     UDP: socket monitoring interface
          < >     RAW: socket monitoring interface
          [ ]     INET: allow privileged process to administratively close sockets
          [*]   TCP: advanced congestion control  --->
          [ ]   TCP: MD5 Signature Option support (RFC2385)
          [ ]   TCP: Transport Layer Compression support
          <*>   The IPv6 protocol  --->
          [ ]   NetLabel subsystem support
          [ ]   MPTCP: Multipath TCP
          -*- Security Marking
          [*] Timestamping in PHY devices
          [*] Network packet filtering framework (Netfilter)  --->
          [*] BPF based packet filtering framework (BPFILTER)  --->
          < > The DCCP Protocol  ----
          < > The SCTP Protocol  ----
```



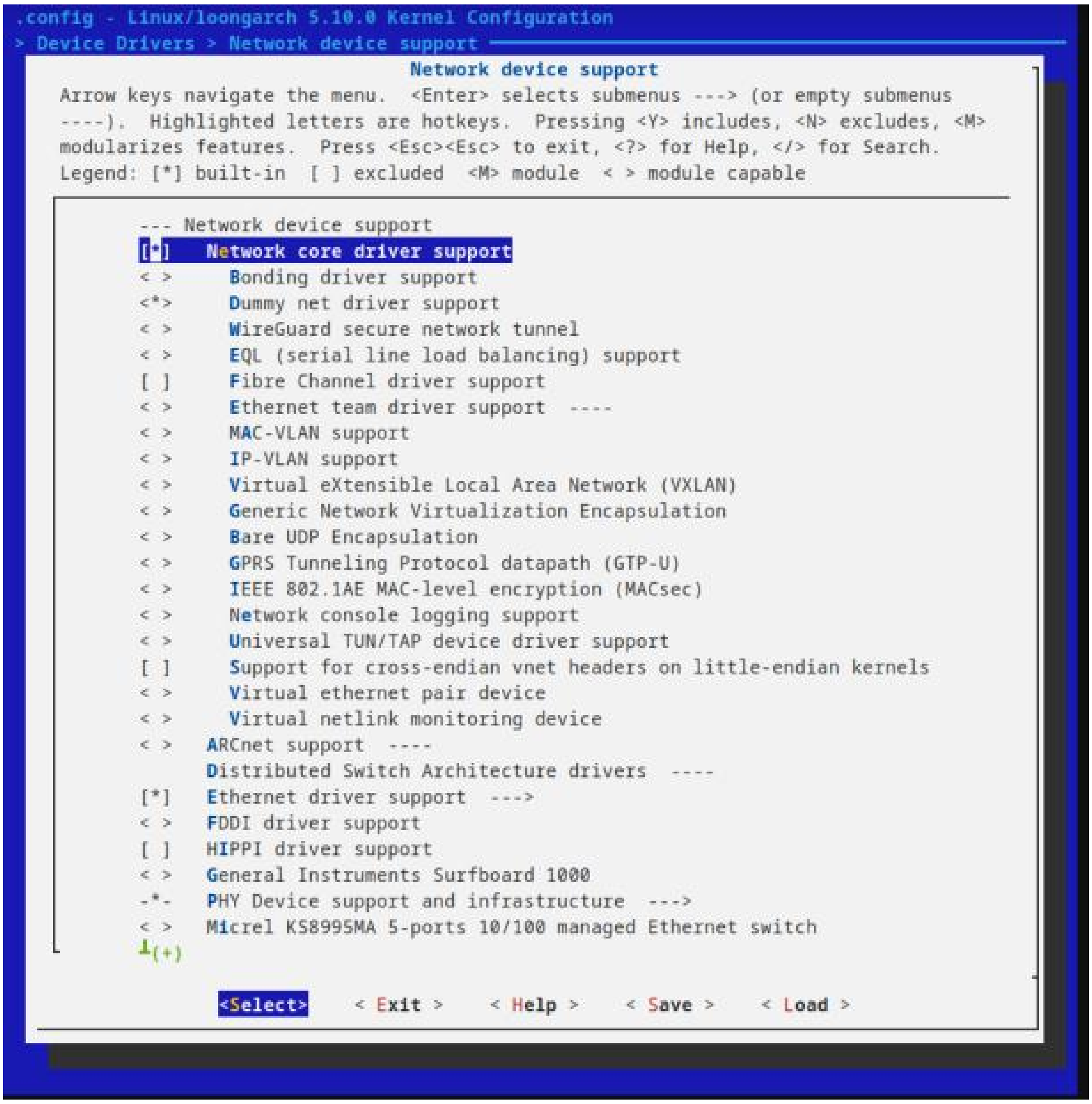
```
    <*> The Reliable Datagram Sockets Protocol
    < >   RDS over TCP
    [ ]   RDS debugging messages
    < > The TIPC Protocol  ----
    < > Asynchronous Transfer Mode (ATM)
    < > Layer Two Tunneling Protocol (L2TP)  ----
    < > 802.1d Ethernet Bridging
    < > Distributed Switch Architecture  ----
    < > 802.1Q/802.1ad VLAN Support
    < > DECnet Support
    < > ANSI/IEEE 802.2 LLC type 2 Support
    < > Appletalk protocol support
    < > CCITT X.25 Packet Layer
    < > LAPB Data Link Driver
    < > Phonet protocols family
    < > 6LoWPAN Support  ----
    < > IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support  ----
    [*] QoS and/or fair queueing  --->
    [*] Data Center Bridging support
    -*- DNS Resolver support
    < > B.A.T.M.A.N. Advanced Meshing Protocol
    < > Open vSwitch
    < > Virtual Socket protocol
    <*> NETLINK: socket monitoring interface
    [ ] MultiProtocol Label Switching  ----
    < > Network Service Header (NSH) protocol  ----
    < > High-availability Seamless Redundancy (HSR & PRP)
    [ ] Switch (and switch-ish) device support
    [ ] L3 Master device support
    < > Qualcomm IPC Router support
    [ ] NCSI interface support
    [*] Network priority cgroup
    [ ] Network classid cgroup
    [*] enable BPF Just In Time compiler
    [ ] enable BPF STREAM_PARSER
    [ ] Network testing  --->

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

配置完成之后，回到主菜单页面，进入“*Device Drivers*”，选中“*Network device support*”，按空格选为“*”，然后按回车 进入。

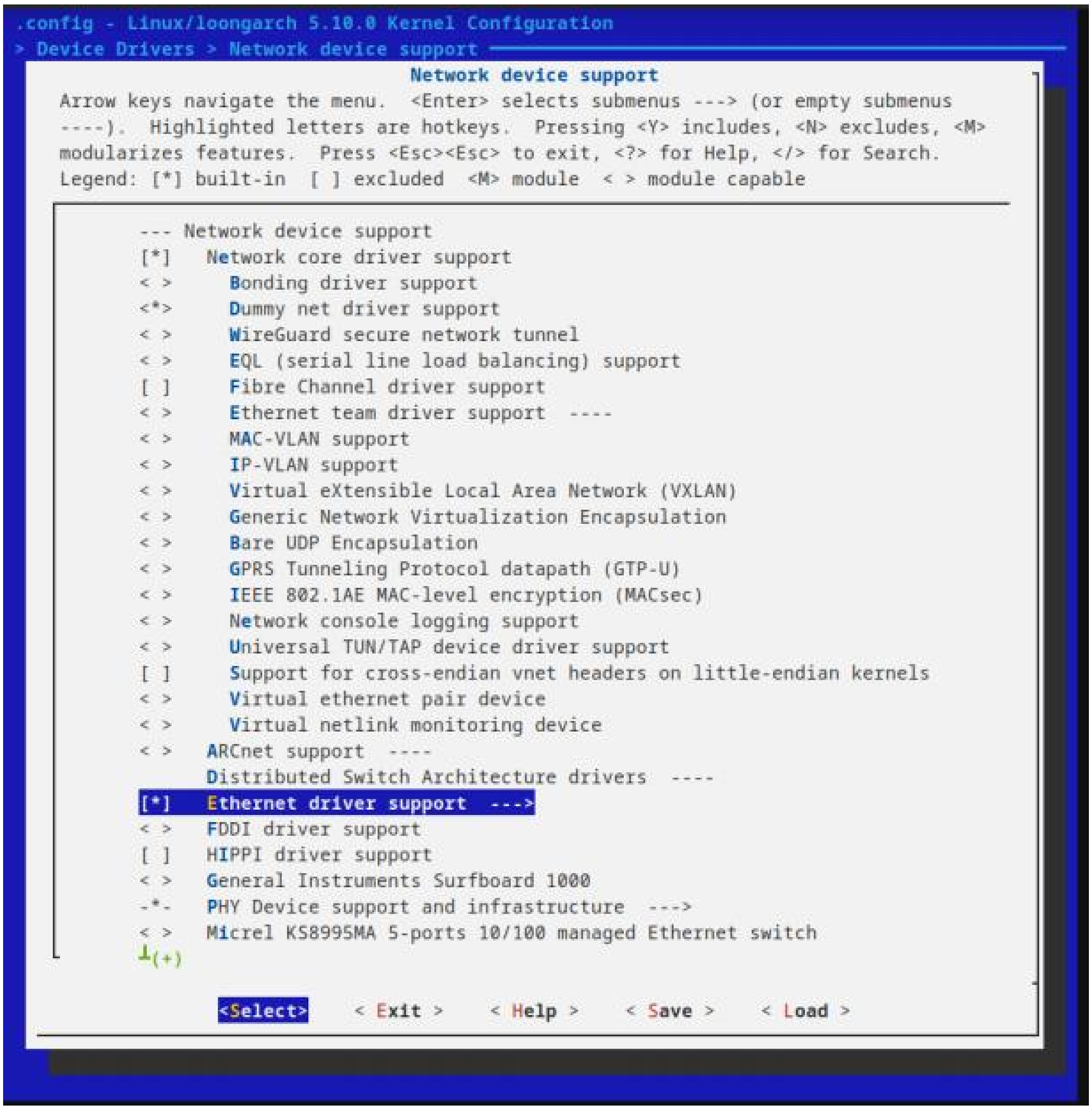


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        -*- PCI support  --->
        < > PCCard (PCMCIA/CardBus) support  ----
        < > RapidIO support  ----
            Generic Driver Options  --->
            Bus devices  --->
        < > Connector - unified userspace <-> kernelspace linker  ----
        < > GNSS receiver support  ----
        <*> Memory Technology Device (MTD) support  --->
        -*- Device Tree and Open Firmware support  --->
        < > Parallel port support  ----
        -*- Plug and Play support  --->
        [*] Block devices  --->
            NVME Support  --->
            Misc devices  --->
            SCSI device support  --->
        <*> Serial ATA and Parallel ATA drivers (libata)  --->
        [ ] Multiple devices driver support (RAID and LVM)  ----
        < > Generic Target Core Mod (TCM) and ConfigFS Infrastructure  ----
        [ ] Fusion MPT device support  ----
            IEEE 1394 (FireWire) support  --->
        [*] Network device support  --->
        [ ] Open-Channel SSD target support  ----
            Input device support  --->
            Character devices  --->
        [ ] Trust the bootloader to initialize Linux's CRNG
            I2C support  --->
        < > I3C support  ----
        [*] SPI support  --->
        ↓(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中"*Network core driver support*"，按空格选择为"*"。然后选中"*Ethernet driver support*"选项，按回车进入

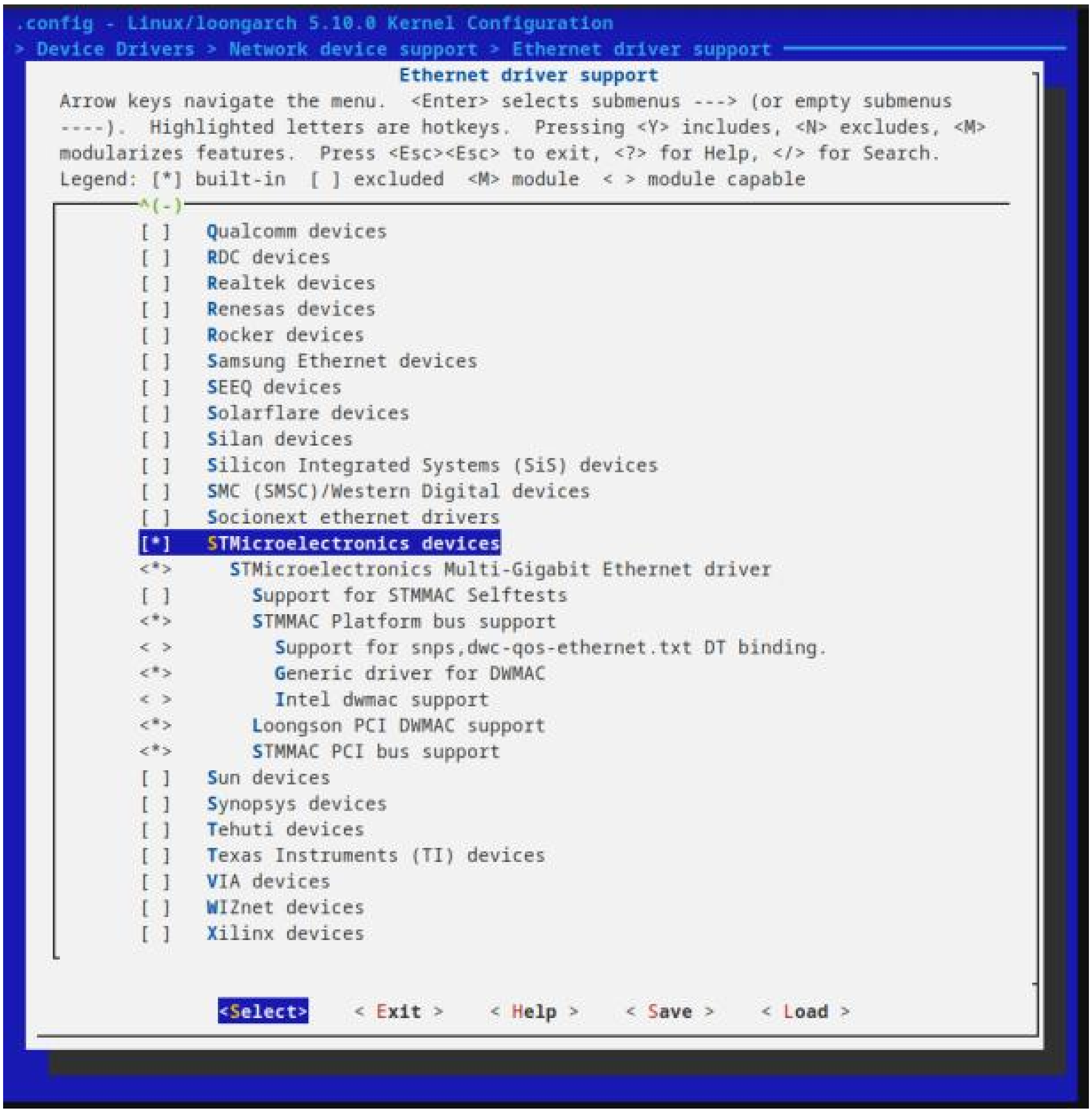


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Network device support ─────────────────────────────────────
                          Network device support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Network device support
        [*]   Network core driver support
        < >     Bonding driver support
        <*>     Dummy net driver support
        < >     WireGuard secure network tunnel
        < >     EQL (serial line load balancing) support
        [ ]     Fibre Channel driver support
        < >     Ethernet team driver support  ----
        < >     MAC-VLAN support
        < >     IP-VLAN support
        < >     Virtual eXtensible Local Area Network (VXLAN)
        < >     Generic Network Virtualization Encapsulation
        < >     Bare UDP Encapsulation
        < >     GPRS Tunneling Protocol datapath (GTP-U)
        < >     IEEE 802.1AE MAC-level encryption (MACsec)
        < >     Network console logging support
        < >     Universal TUN/TAP device driver support
        [ ]     Support for cross-endian vnet headers on little-endian kernels
        < >     Virtual ethernet pair device
        < >     Virtual netlink monitoring device
        < >   ARCnet support  ----
              Distributed Switch Architecture drivers  ----
        [*]   Ethernet driver support  --->
        < >   FDDI driver support
        [ ]   HIPPI driver support
        < >   General Instruments Surfboard 1000
        -*-   PHY Device support and infrastructure  --->
        < >   Micrel KS8995MA 5-ports 10/100 managed Ethernet switch
        ┴(+)

            <Select>    < Exit >    < Help >    < Save >    < Load >
```



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Network device support
                         Network device support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Network device support
        [*]   Network core driver support
        < >     Bonding driver support
        <*>     Dummy net driver support
        < >     WireGuard secure network tunnel
        < >     EQL (serial line load balancing) support
        [ ]     Fibre Channel driver support
        < >     Ethernet team driver support  ----
        < >     MAC-VLAN support
        < >     IP-VLAN support
        < >     Virtual eXtensible Local Area Network (VXLAN)
        < >     Generic Network Virtualization Encapsulation
        < >     Bare UDP Encapsulation
        < >     GPRS Tunneling Protocol datapath (GTP-U)
        < >     IEEE 802.1AE MAC-level encryption (MACsec)
        < >     Network console logging support
        < >     Universal TUN/TAP device driver support
        [ ]     Support for cross-endian vnet headers on little-endian kernels
        < >     Virtual ethernet pair device
        < >     Virtual netlink monitoring device
        < >   ARCnet support  ----
              Distributed Switch Architecture drivers  ----
        [*]   Ethernet driver support  --->
        < >   FDDI driver support
        [ ]   HIPPI driver support
        < >   General Instruments Surfboard 1000
        -*-   PHY Device support and infrastructure  --->
        < >   Micrel KS8995MA 5-ports 10/100 managed Ethernet switch
        ↓(+)

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中以下选项，并且按回车选为“*”。

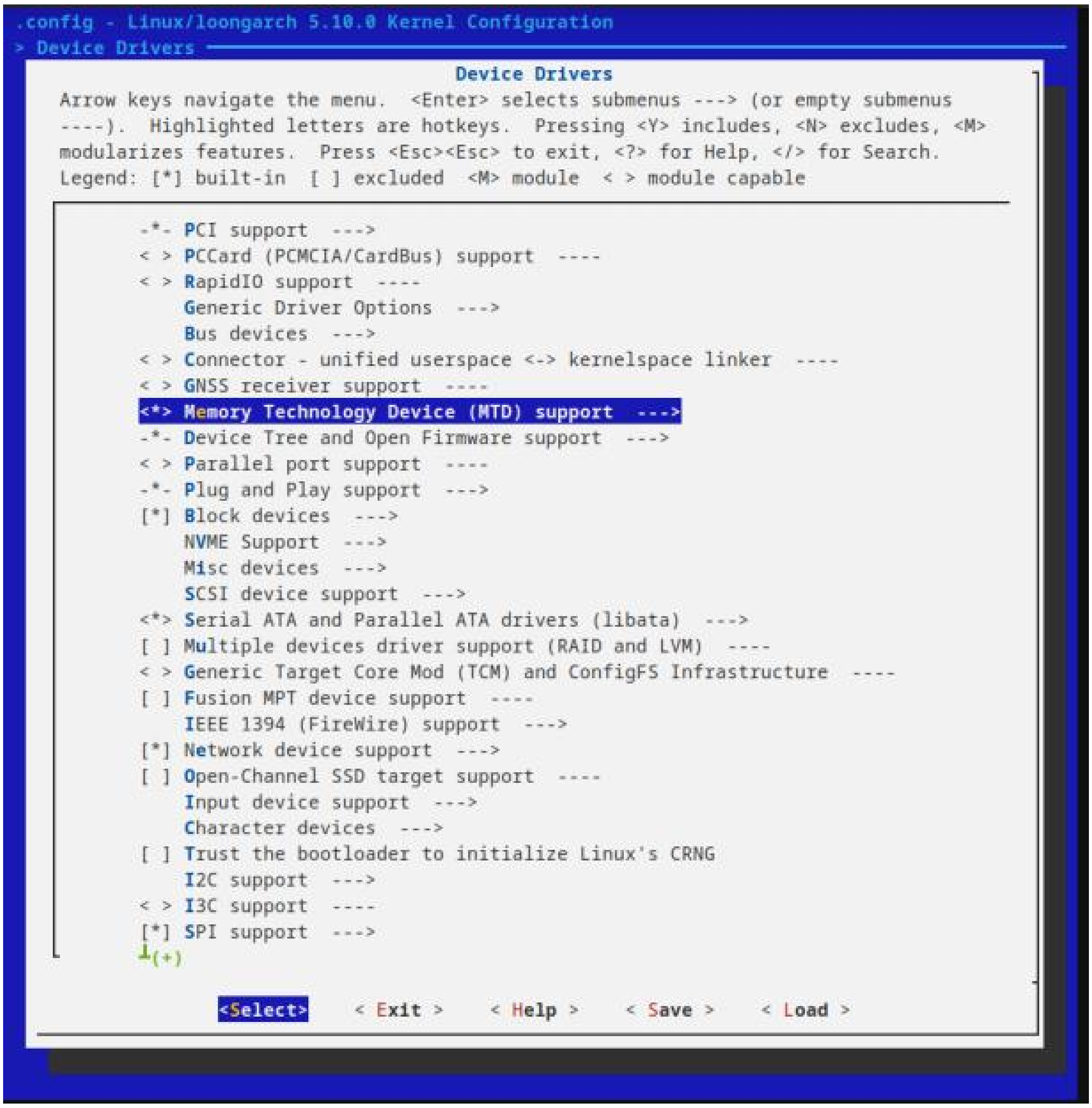


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Network device support > Ethernet driver support ──────────
                         Ethernet driver support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        ^(-)
        [ ]   Qualcomm devices
        [ ]   RDC devices
        [ ]   Realtek devices
        [ ]   Renesas devices
        [ ]   Rocker devices
        [ ]   Samsung Ethernet devices
        [ ]   SEEQ devices
        [ ]   Solarflare devices
        [ ]   Silan devices
        [ ]   Silicon Integrated Systems (SiS) devices
        [ ]   SMC (SMSC)/Western Digital devices
        [ ]   Socionext ethernet drivers
        [*]   STMicroelectronics devices
        <*>     STMicroelectronics Multi-Gigabit Ethernet driver
        [ ]       Support for STMMAC Selftests
        <*>       STMMAC Platform bus support
        < >         Support for snps,dwc-qos-ethernet.txt DT binding.
        <*>         Generic driver for DWMAC
        < >         Intel dwmac support
        <*>       Loongson PCI DWMAC support
        <*>       STMMAC PCI bus support
        [ ]   Sun devices
        [ ]   Synopsys devices
        [ ]   Tehuti devices
        [ ]   Texas Instruments (TI) devices
        [ ]   VIA devices
        [ ]   WIZnet devices
        [ ]   Xilinx devices

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

10.2.9.NAND驱动

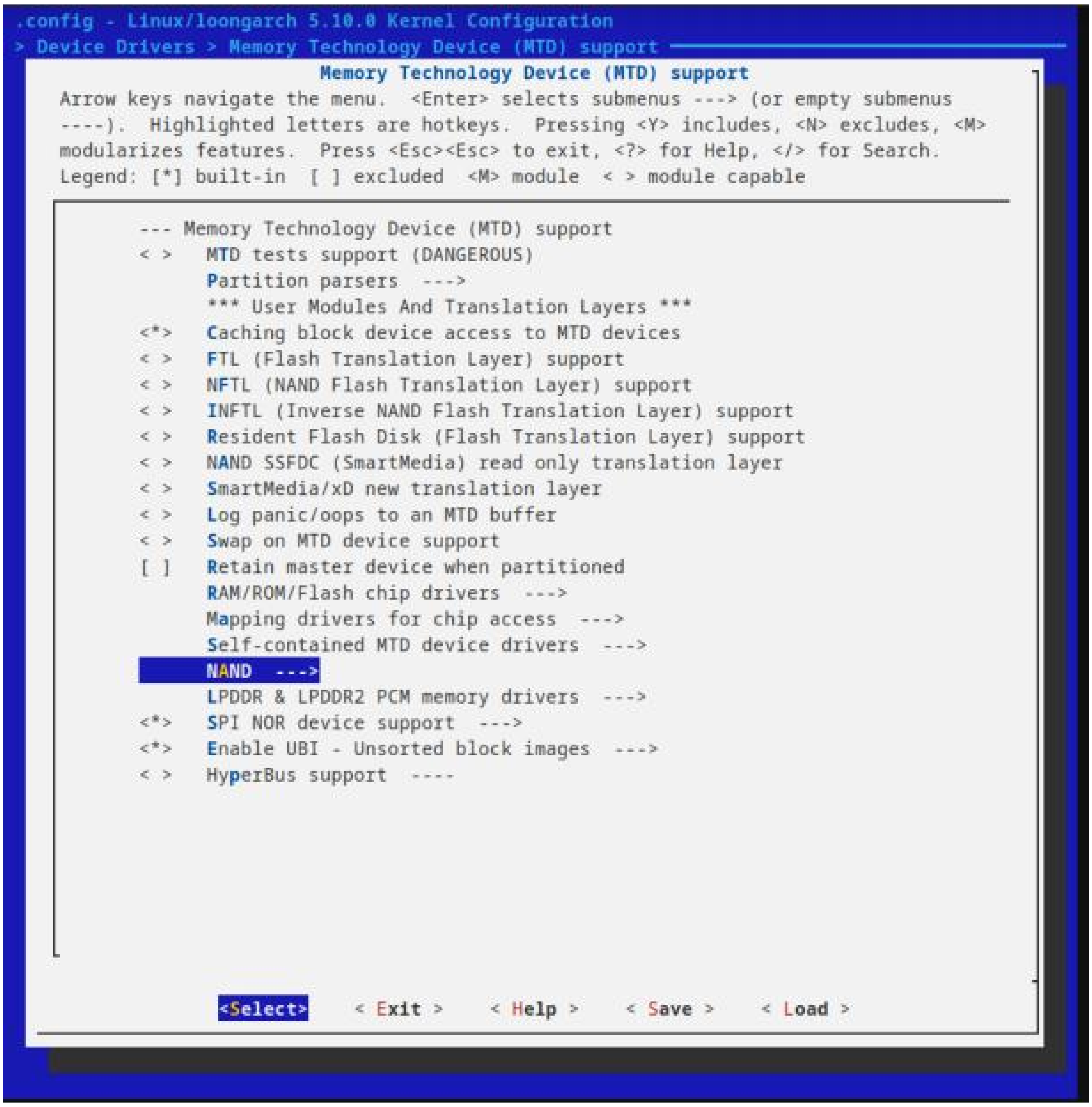
NAND上作为一个系统盘，是当作MTD。当在uboot中选择为在NAND中启动系统。 uboot传递给内核（NAND中）的引 导参数中关于文件系统的信息则会改为在NAND中的文件系统的信息。

Device Drivers 选中“Memory Technology Device (MTD) support”，按空格选为“*”，按回车进入。

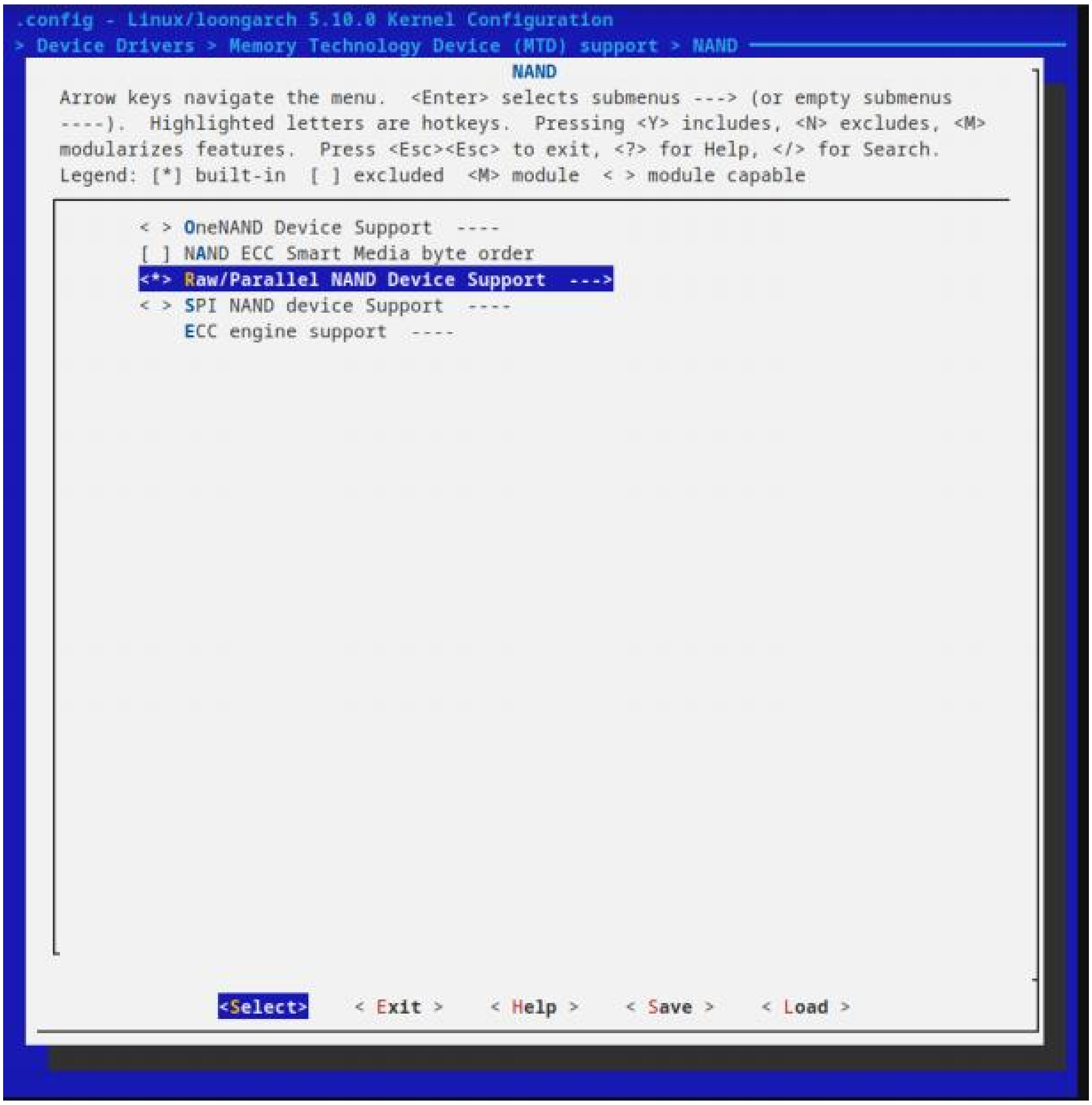


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        -*- PCI support  --->
        < > PCCard (PCMCIA/CardBus) support  ----
        < > RapidIO support  ----
            Generic Driver Options  --->
            Bus devices  --->
        < > Connector - unified userspace <-> kernelspace linker  ----
        < > GNSS receiver support  ----
        <*> Memory Technology Device (MTD) support  --->
        -*- Device Tree and Open Firmware support  --->
        < > Parallel port support  ----
        -*- Plug and Play support  --->
        [*] Block devices  --->
            NVME Support  --->
            Misc devices  --->
            SCSI device support  --->
        <*> Serial ATA and Parallel ATA drivers (libata)  --->
        [ ] Multiple devices driver support (RAID and LVM)  ----
        < > Generic Target Core Mod (TCM) and ConfigFS Infrastructure  ----
        [ ] Fusion MPT device support  ----
            IEEE 1394 (FireWire) support  --->
        [*] Network device support  --->
        [ ] Open-Channel SSD target support  ----
            Input device support  --->
            Character devices  --->
        [ ] Trust the bootloader to initialize Linux's CRNG
            I2C support  --->
        < > I3C support  ----
        [*] SPI support  --->
        ↓(+)

             <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中“NAND”->“Raw/Parallel NAND Device Support”，按空格选为“*”，按回车进入。

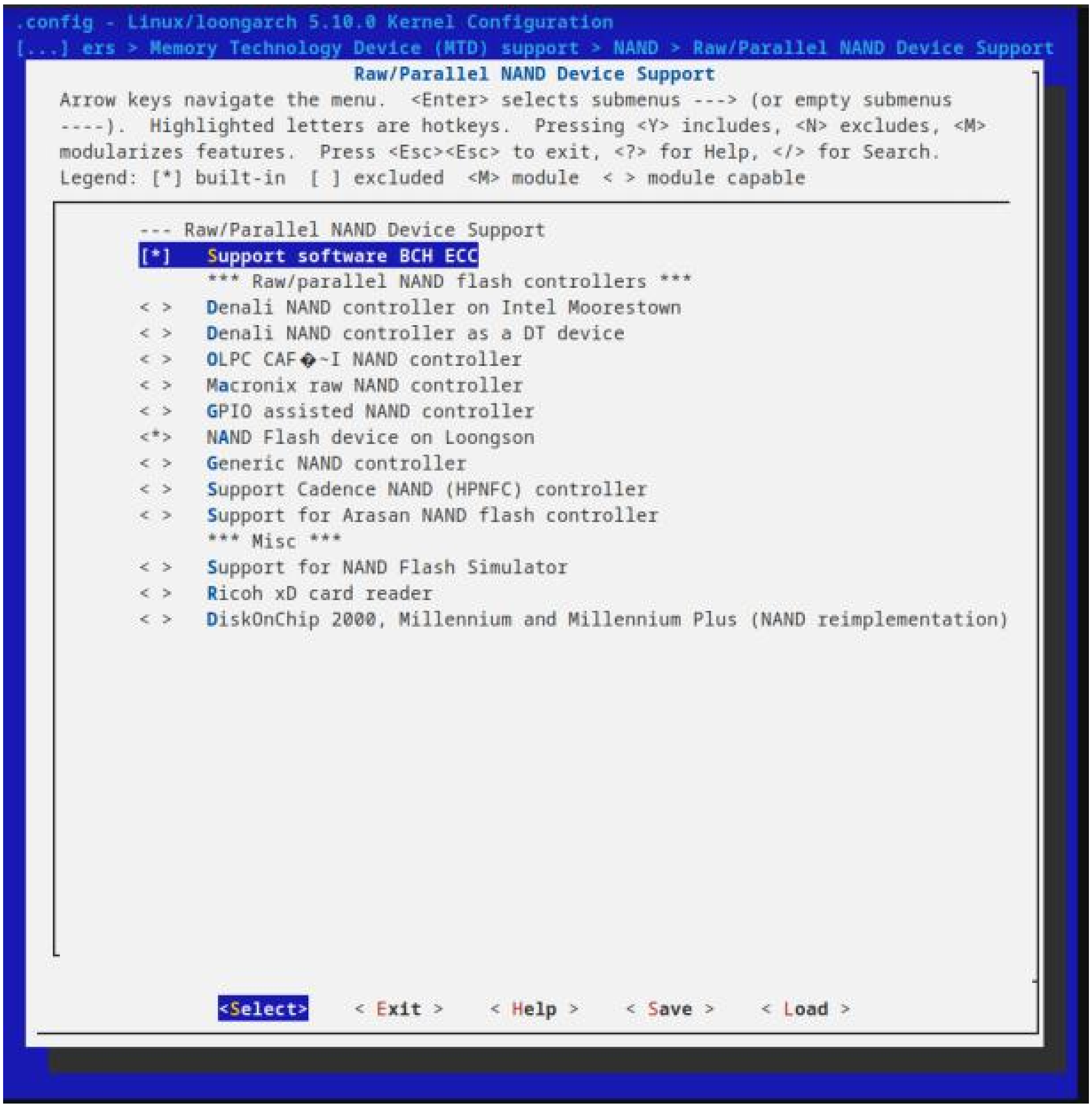


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Memory Technology Device (MTD) support
                    Memory Technology Device (MTD) support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Memory Technology Device (MTD) support
        < >   MTD tests support (DANGEROUS)
              Partition parsers  --->
              *** User Modules And Translation Layers ***
        <*>   Caching block device access to MTD devices
        < >   FTL (Flash Translation Layer) support
        < >   NFTL (NAND Flash Translation Layer) support
        < >   INFTL (Inverse NAND Flash Translation Layer) support
        < >   Resident Flash Disk (Flash Translation Layer) support
        < >   NAND SSFDC (SmartMedia) read only translation layer
        < >   SmartMedia/xD new translation layer
        < >   Log panic/oops to an MTD buffer
        < >   Swap on MTD device support
        [ ]   Retain master device when partitioned
              RAM/ROM/Flash chip drivers  --->
              Mapping drivers for chip access  --->
              Self-contained MTD device drivers  --->
              NAND  --->
              LPDDR & LPDDR2 PCM memory drivers  --->
        <*>   SPI NOR device support  --->
        <*>   Enable UBI - Unsorted block images  --->
        < >   HyperBus support  ----

              <Select>    < Exit >    < Help >    < Save >    < Load >
```

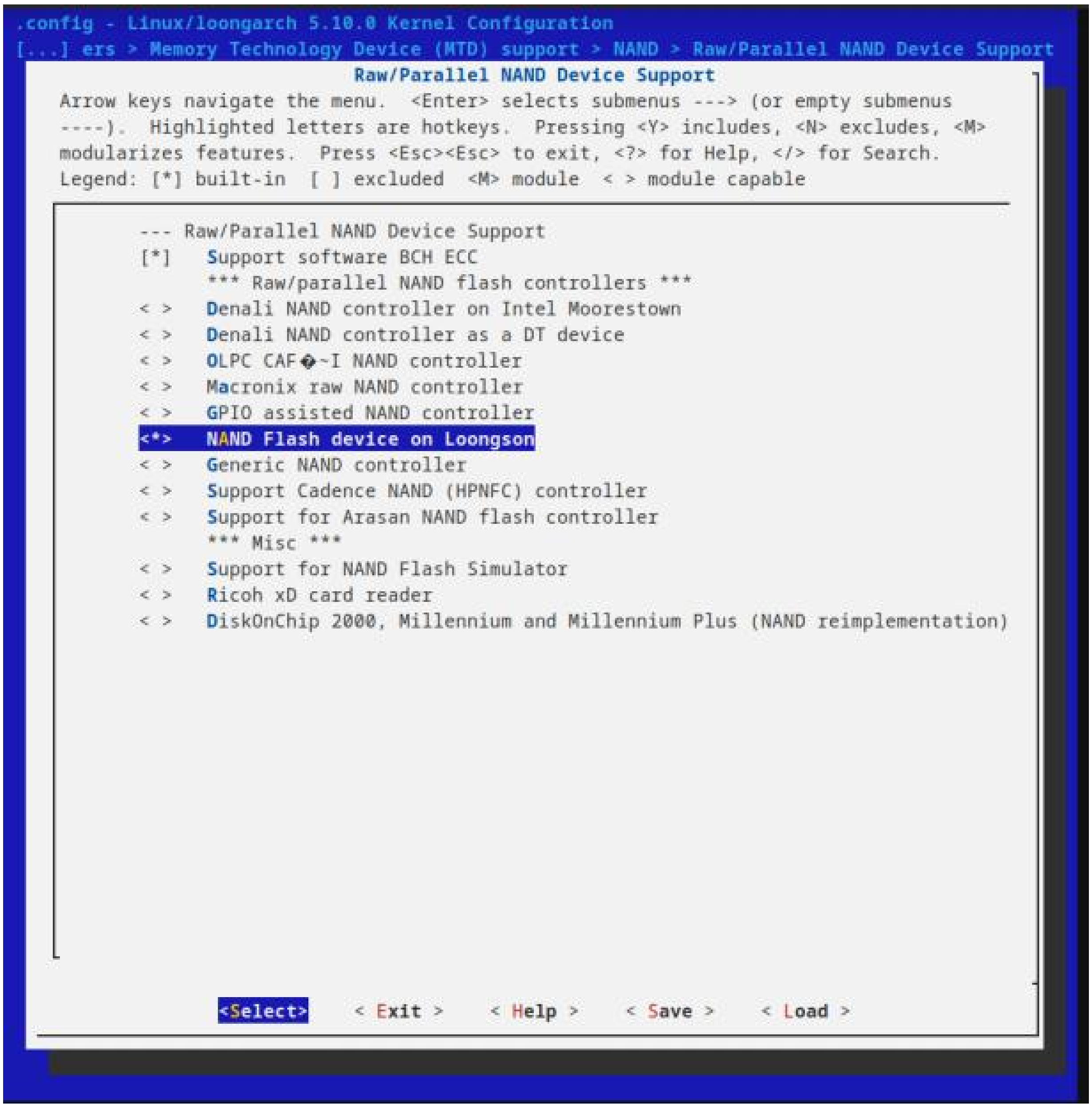


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Memory Technology Device (MTD) support > NAND
                                  NAND
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        < > OneNAND Device Support  ----
        [ ] NAND ECC Smart Media byte order
        <*> Raw/Parallel NAND Device Support  --->
        < > SPI NAND device Support  ----
            ECC engine support  ----

            <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中“Support software BCH ECC”和“Support for NAND flash devices on Loongson”选项，按空格选为“*”



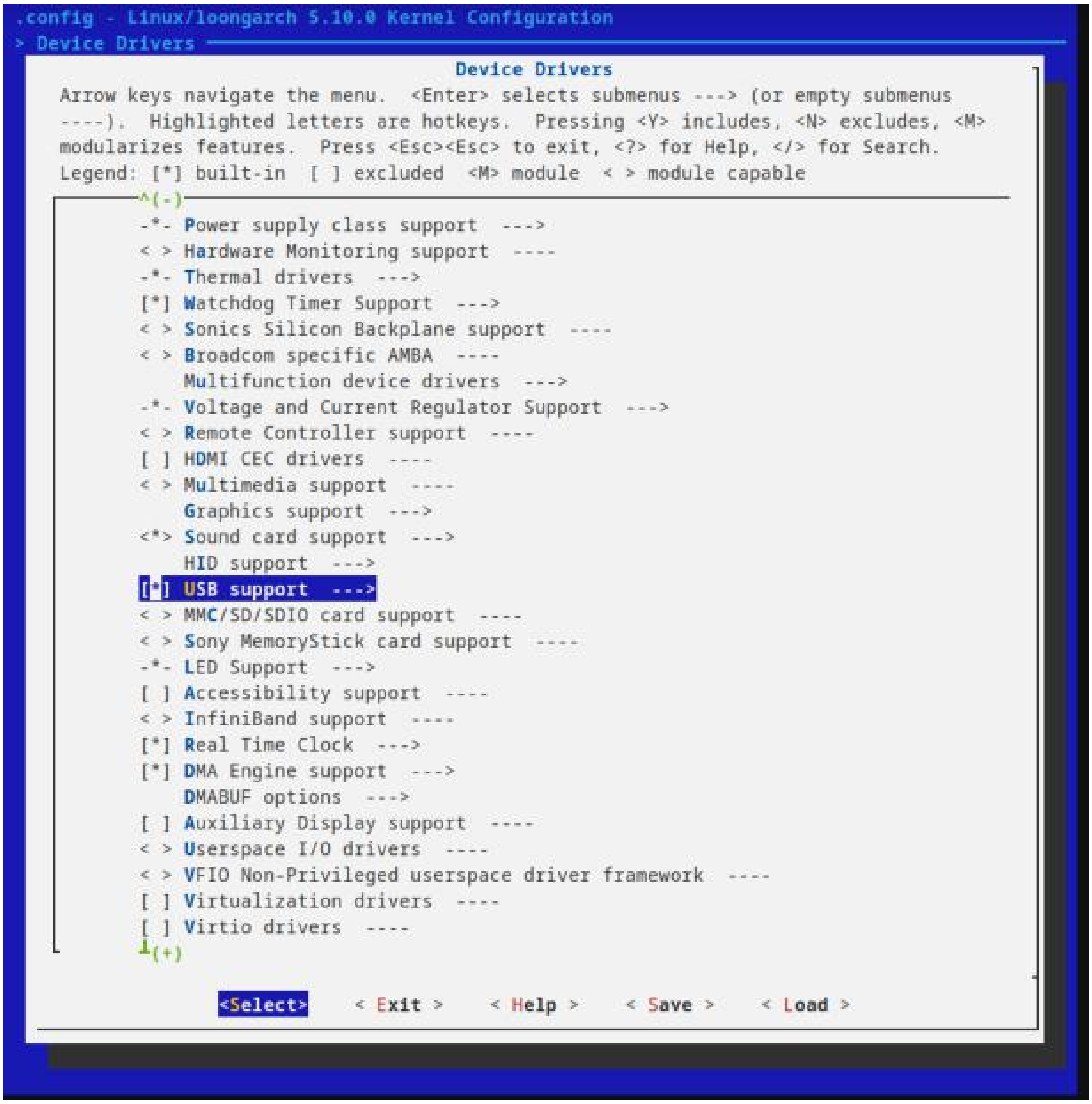
```
.config - Linux/loongarch 5.10.0 Kernel Configuration
[...] ers > Memory Technology Device (MTD) support > NAND > Raw/Parallel NAND Device Support
                            Raw/Parallel NAND Device Support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Raw/Parallel NAND Device Support
        [*]   Support software BCH ECC
              *** Raw/parallel NAND flash controllers ***
        < >   Denali NAND controller on Intel Moorestown
        < >   Denali NAND controller as a DT device
        < >   OLPC CAF�~I NAND controller
        < >   Macronix raw NAND controller
        < >   GPIO assisted NAND controller
        <*>   NAND Flash device on Loongson
        < >   Generic NAND controller
        < >   Support Cadence NAND (HPNFC) controller
        < >   Support for Arasan NAND flash controller
              *** Misc ***
        < >   Support for NAND Flash Simulator
        < >   Ricoh xD card reader
        < >   DiskOnChip 2000, Millennium and Millennium Plus (NAND reimplementation)

               <Select>    < Exit >    < Help >    < Save >    < Load >
```



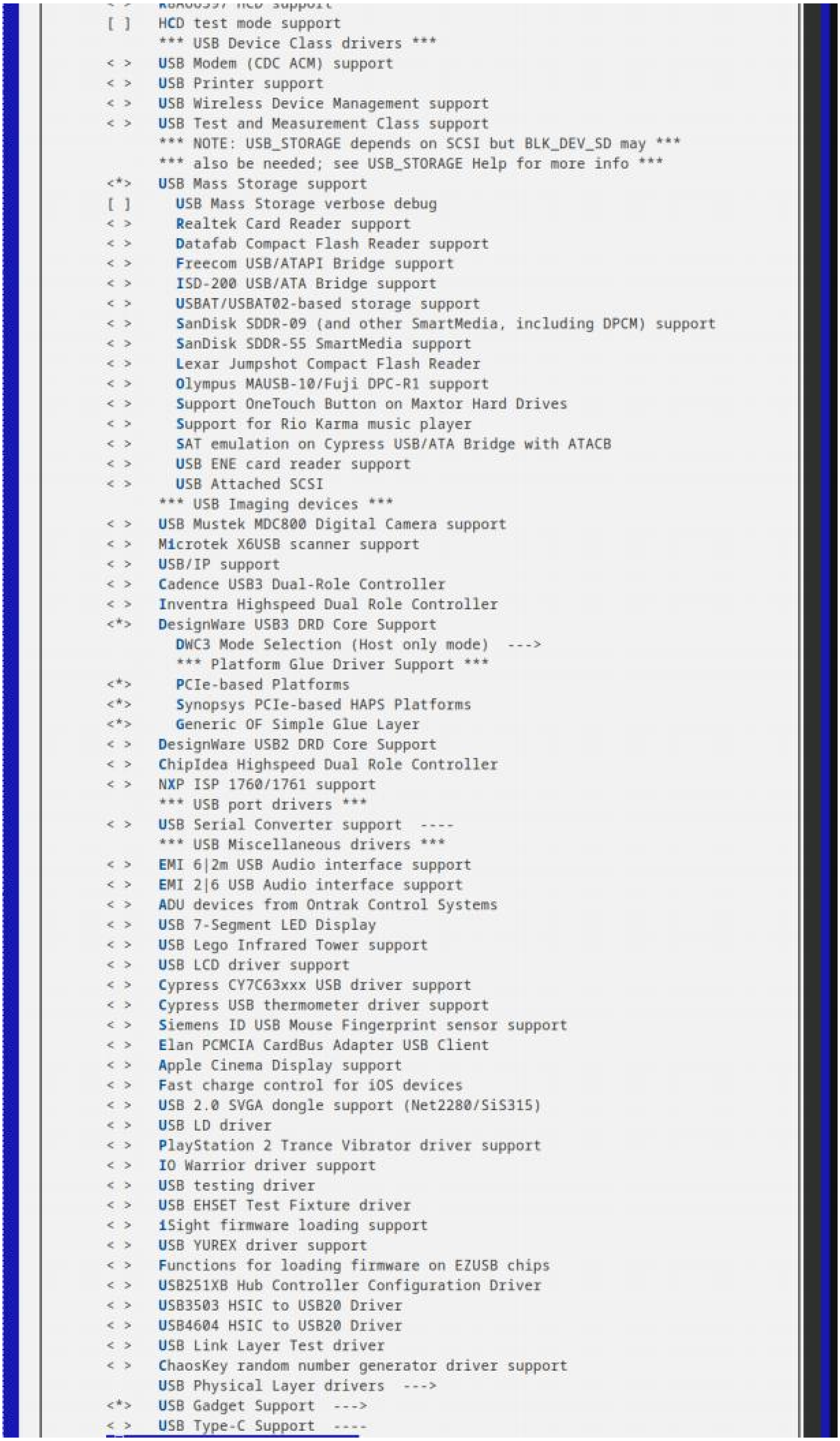
```
.config - Linux/loongarch 5.10.0 Kernel Configuration
[...] ers > Memory Technology Device (MTD) support > NAND > Raw/Parallel NAND Device Support
                        Raw/Parallel NAND Device Support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Raw/Parallel NAND Device Support
        [*]   Support software BCH ECC
              *** Raw/parallel NAND flash controllers ***
        < >   Denali NAND controller on Intel Moorestown
        < >   Denali NAND controller as a DT device
        < >   OLPC CAF�~I NAND controller
        < >   Macronix raw NAND controller
        < >   GPIO assisted NAND controller
        <*>   NAND Flash device on Loongson
        < >   Generic NAND controller
        < >   Support Cadence NAND (HPNFC) controller
        < >   Support for Arasan NAND flash controller
              *** Misc ***
        < >   Support for NAND Flash Simulator
        < >   Ricoh xD card reader
        < >   DiskOnChip 2000, Millennium and Millennium Plus (NAND reimplementation)

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

10.2.10. USB驱动

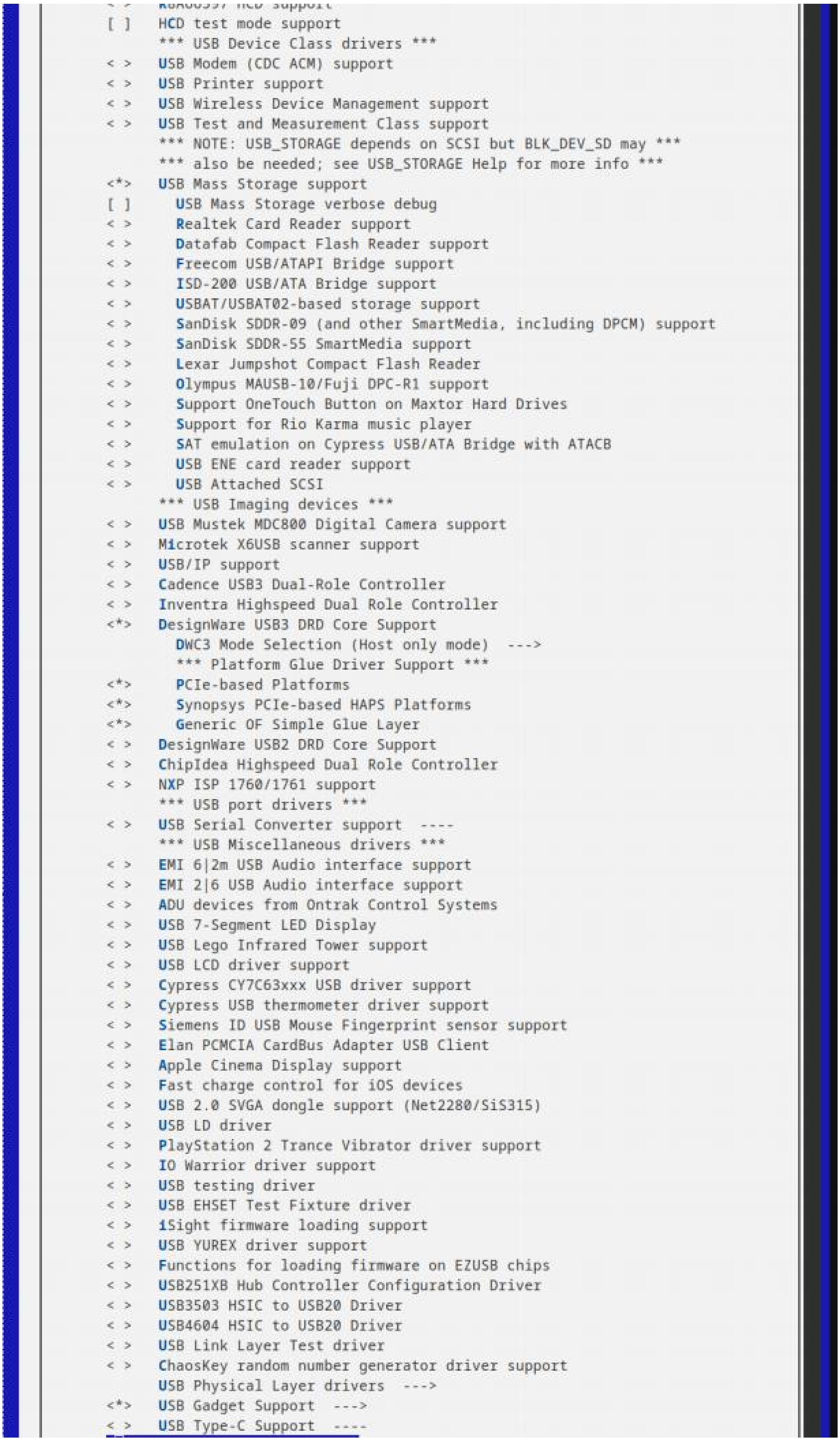
Device Drivers 选中“USB support”按回车进入。


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers ──────────────────────────────────────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
 ┌──────^(-)──────────────────────────────────────────────────────────────────┐
 │       -*- Power supply class support  --->
 │       < > Hardware Monitoring support  ----
 │       -*- Thermal drivers  --->
 │       [*] Watchdog Timer Support  --->
 │       < > Sonics Silicon Backplane support  ----
 │       < > Broadcom specific AMBA  ----
 │           Multifunction device drivers  --->
 │       -*- Voltage and Current Regulator Support  --->
 │       < > Remote Controller support  ----
 │       [ ] HDMI CEC drivers  ----
 │       < > Multimedia support  ----
 │           Graphics support  --->
 │       <*> Sound card support  --->
 │           HID support  --->
 │       [*] USB support  --->
 │       < > MMC/SD/SDIO card support  ----
 │       < > Sony MemoryStick card support  ----
 │       -*- LED Support  --->
 │       [ ] Accessibility support  ----
 │       < > InfiniBand support  ----
 │       [*] Real Time Clock  --->
 │       [*] DMA Engine support  --->
 │           DMABUF options  --->
 │       [ ] Auxiliary Display support  ----
 │       < > Userspace I/O drivers  ----
 │       < > VFIO Non-Privileged userspace driver framework  ----
 │       [ ] Virtualization drivers  ----
 │       [ ] Virtio drivers  ----
 └───────┴(+)─────────────────────────────────────────────────────────────────┘

              <Select>    < Exit >    < Help >    < Save >    < Load >
```

按以下选项勾选



```
< >   R8A66597 HCD support
[ ]   HCD test mode support
      *** USB Device Class drivers ***
< >   USB Modem (CDC ACM) support
< >   USB Printer support
< >   USB Wireless Device Management support
< >   USB Test and Measurement Class support
      *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
      *** also be needed; see USB_STORAGE Help for more info ***
<*>   USB Mass Storage support
[ ]     USB Mass Storage verbose debug
< >     Realtek Card Reader support
< >     Datafab Compact Flash Reader support
< >     Freecom USB/ATAPI Bridge support
< >     ISD-200 USB/ATA Bridge support
< >     USBAT/USBAT02-based storage support
< >     SanDisk SDDR-09 (and other SmartMedia, including DPCM) support
< >     SanDisk SDDR-55 SmartMedia support
< >     Lexar Jumpshot Compact Flash Reader
< >     Olympus MAUSB-10/Fuji DPC-R1 support
< >     Support OneTouch Button on Maxtor Hard Drives
< >     Support for Rio Karma music player
< >     SAT emulation on Cypress USB/ATA Bridge with ATACB
< >     USB ENE card reader support
< >     USB Attached SCSI
      *** USB Imaging devices ***
< >   USB Mustek MDC800 Digital Camera support
< >   Microtek X6USB scanner support
< >   USB/IP support
< >   Cadence USB3 Dual-Role Controller
< >   Inventra Highspeed Dual Role Controller
<*>   DesignWare USB3 DRD Core Support
        DWC3 Mode Selection (Host only mode)  --->
        *** Platform Glue Driver Support ***
<*>     PCIe-based Platforms
<*>     Synopsys PCIe-based HAPS Platforms
<*>     Generic OF Simple Glue Layer
< >   DesignWare USB2 DRD Core Support
< >   ChipIdea Highspeed Dual Role Controller
< >   NXP ISP 1760/1761 support
      *** USB port drivers ***
< >   USB Serial Converter support  ----
      *** USB Miscellaneous drivers ***
< >   EMI 6|2m USB Audio interface support
< >   EMI 2|6 USB Audio interface support
< >   ADU devices from Ontrak Control Systems
< >   USB 7-Segment LED Display
< >   USB Lego Infrared Tower support
< >   USB LCD driver support
< >   Cypress CY7C63xxx USB driver support
< >   Cypress USB thermometer driver support
< >   Siemens ID USB Mouse Fingerprint sensor support
< >   Elan PCMCIA CardBus Adapter USB Client
< >   Apple Cinema Display support
< >   Fast charge control for iOS devices
< >   USB 2.0 SVGA dongle support (Net2280/SiS315)
< >   USB LD driver
< >   PlayStation 2 Trance Vibrator driver support
< >   IO Warrior driver support
< >   USB testing driver
< >   USB EHSET Test Fixture driver
< >   iSight firmware loading support
< >   USB YUREX driver support
< >   Functions for loading firmware on EZUSB chips
< >   USB251XB Hub Controller Configuration Driver
< >   USB3503 HSIC to USB20 Driver
< >   USB4604 HSIC to USB20 Driver
< >   USB Link Layer Test driver
< >   ChaosKey random number generator driver support
      USB Physical Layer drivers  --->
<*>   USB Gadget Support  --->
< >   USB Type-C Support  ----
```

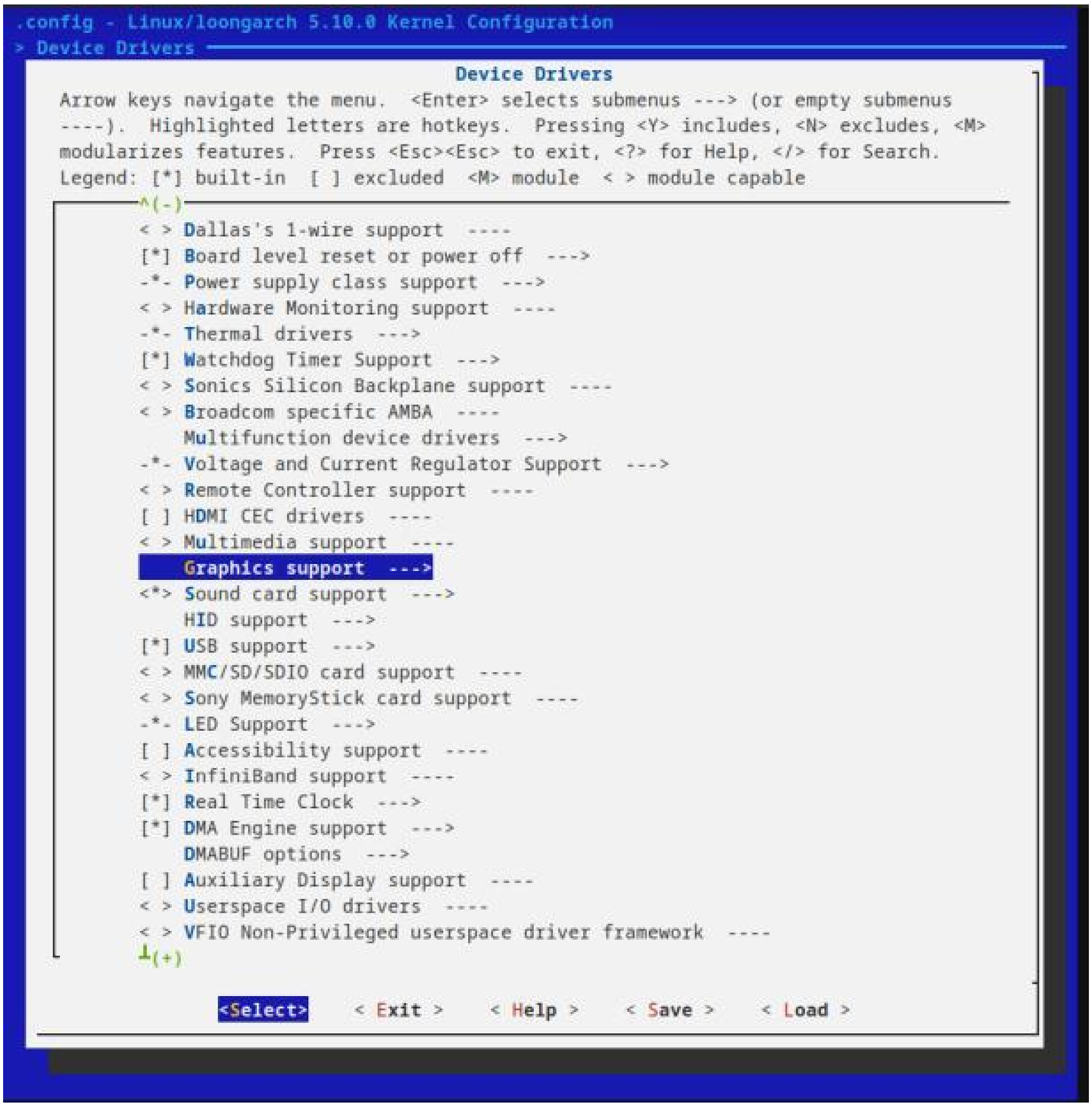


```
     [ ]   HCD test mode support
           *** USB Device Class drivers ***
     < >   USB Modem (CDC ACM) support
     < >   USB Printer support
     < >   USB Wireless Device Management support
     < >   USB Test and Measurement Class support
           *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
           *** also be needed; see USB_STORAGE Help for more info ***
     <*>   USB Mass Storage support
     [ ]     USB Mass Storage verbose debug
     < >     Realtek Card Reader support
     < >     Datafab Compact Flash Reader support
     < >     Freecom USB/ATAPI Bridge support
     < >     ISD-200 USB/ATA Bridge support
     < >     USBAT/USBAT02-based storage support
     < >     SanDisk SDDR-09 (and other SmartMedia, including DPCM) support
     < >     SanDisk SDDR-55 SmartMedia support
     < >     Lexar Jumpshot Compact Flash Reader
     < >     Olympus MAUSB-10/Fuji DPC-R1 support
     < >     Support OneTouch Button on Maxtor Hard Drives
     < >     Support for Rio Karma music player
     < >     SAT emulation on Cypress USB/ATA Bridge with ATACB
     < >     USB ENE card reader support
     < >     USB Attached SCSI
           *** USB Imaging devices ***
     < >   USB Mustek MDC800 Digital Camera support
     < >   Microtek X6USB scanner support
     < >   USB/IP support
     < >   Cadence USB3 Dual-Role Controller
     < >   Inventra Highspeed Dual Role Controller
     <*>   DesignWare USB3 DRD Core Support
             DWC3 Mode Selection (Host only mode)  --->
             *** Platform Glue Driver Support ***
     <*>     PCIe-based Platforms
     <*>     Synopsys PCIe-based HAPS Platforms
     <*>     Generic OF Simple Glue Layer
     < >   DesignWare USB2 DRD Core Support
     < >   ChipIdea Highspeed Dual Role Controller
     < >   NXP ISP 1760/1761 support
           *** USB port drivers ***
     < >   USB Serial Converter support  ----
           *** USB Miscellaneous drivers ***
     < >   EMI 6|2m USB Audio interface support
     < >   EMI 2|6 USB Audio interface support
     < >   ADU devices from Ontrak Control Systems
     < >   USB 7-Segment LED Display
     < >   USB Lego Infrared Tower support
     < >   USB LCD driver support
     < >   Cypress CY7C63xxx USB driver support
     < >   Cypress USB thermometer driver support
     < >   Siemens ID USB Mouse Fingerprint sensor support
     < >   Elan PCMCIA CardBus Adapter USB Client
     < >   Apple Cinema Display support
     < >   Fast charge control for iOS devices
     < >   USB 2.0 SVGA dongle support (Net2280/SiS315)
     < >   USB LD driver
     < >   PlayStation 2 Trance Vibrator driver support
     < >   IO Warrior driver support
     < >   USB testing driver
     < >   USB EHSET Test Fixture driver
     < >   iSight firmware loading support
     < >   USB YUREX driver support
     < >   Functions for loading firmware on EZUSB chips
     < >   USB251XB Hub Controller Configuration Driver
     < >   USB3503 HSIC to USB20 Driver
     < >   USB4604 HSIC to USB20 Driver
     < >   USB Link Layer Test driver
     < >   ChaosKey random number generator driver support
           USB Physical Layer drivers  --->
     <*>   USB Gadget Support  --->
     < >   USB Type-C Support  ----
```

```
        < >   USB Role Switch Support

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

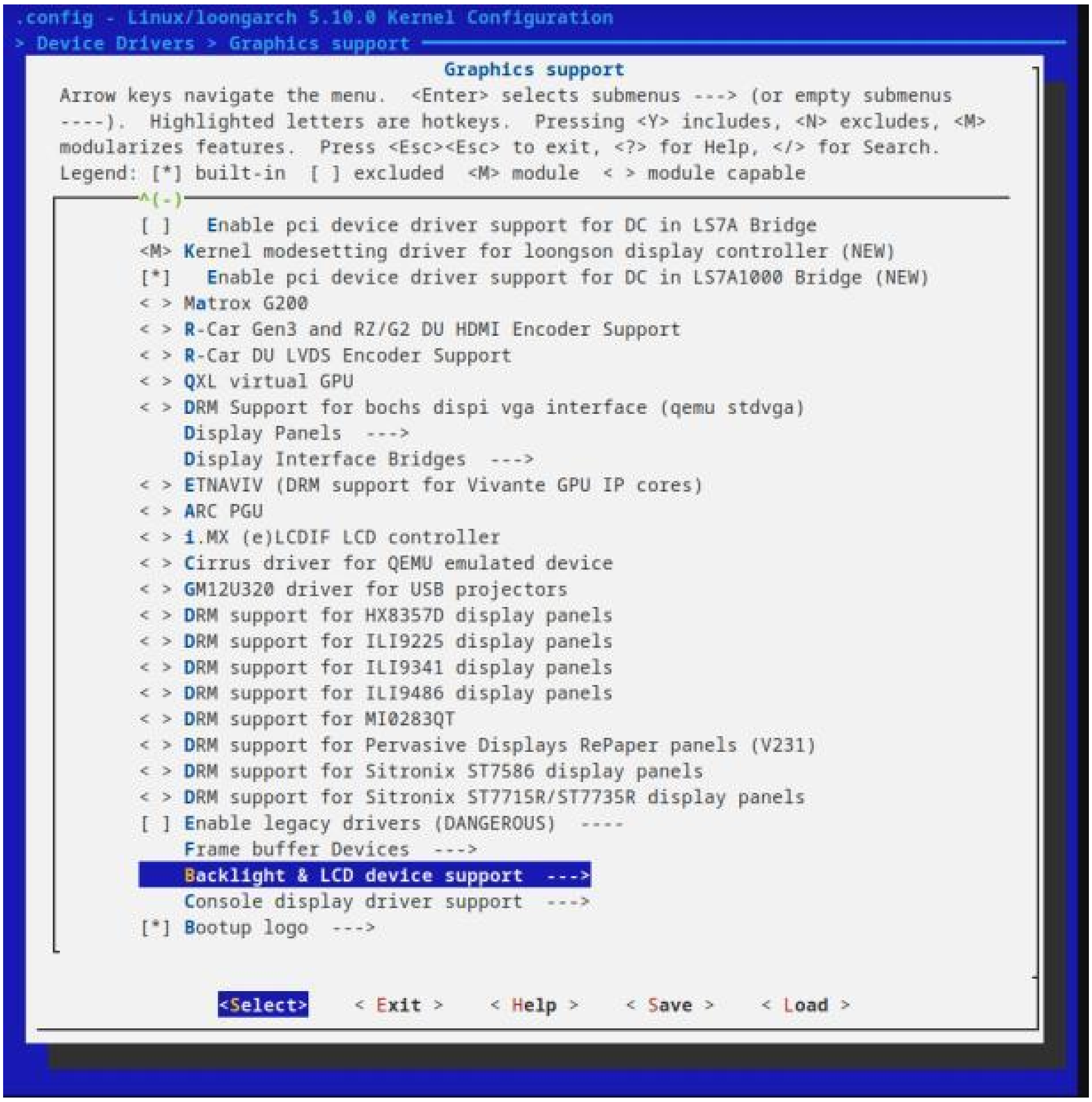
10.2.11. LCD驱动

Device Drivers 选中“Graphics support”，按回车进入。

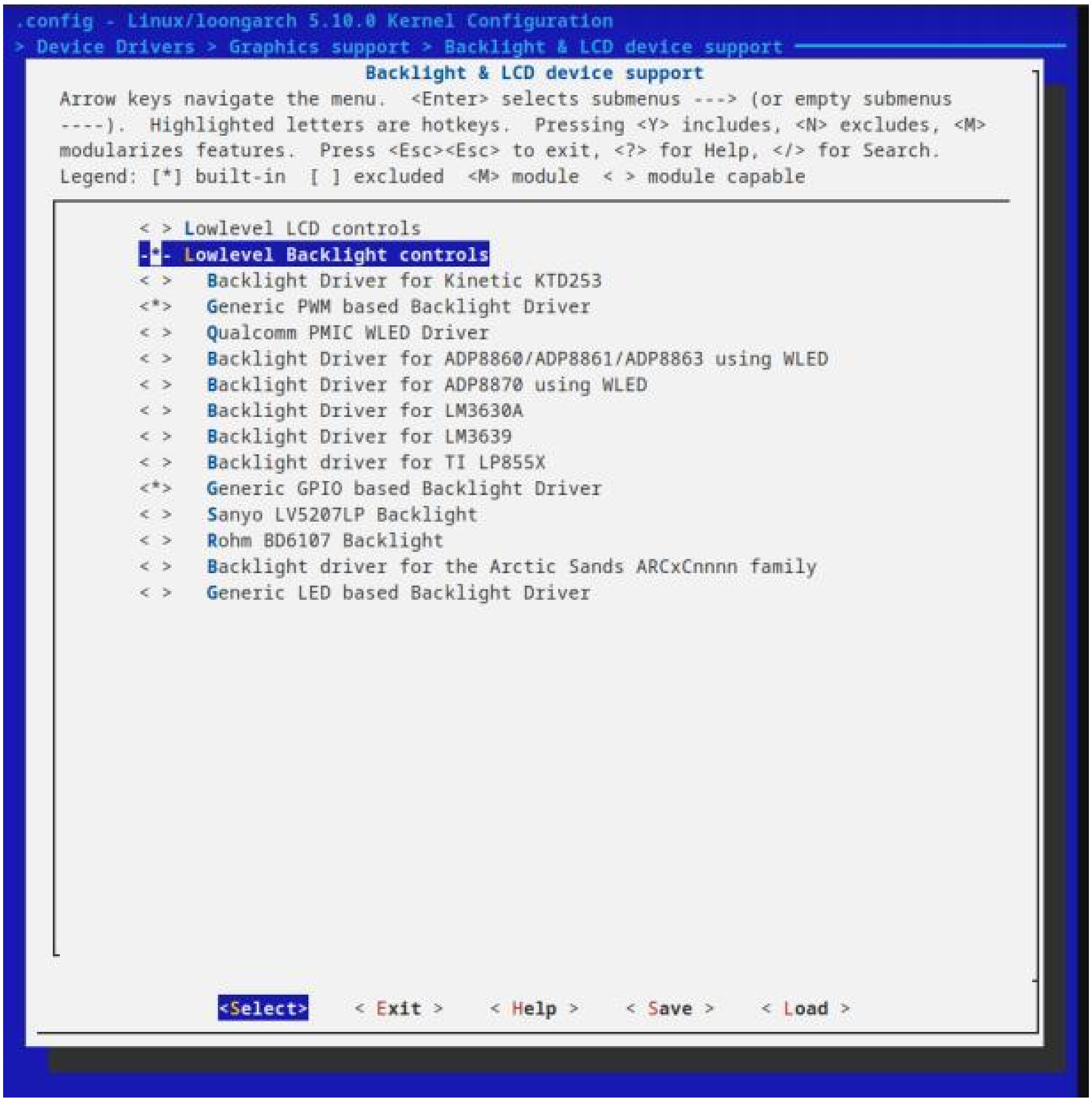


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers ──────────────────────────────────────────────────────
                              Device Drivers
   Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
   ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
   modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
   Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
        ^(-)
        < > Dallas's 1-wire support  ----
        [*] Board level reset or power off  --->
        -*- Power supply class support  --->
        < > Hardware Monitoring support  ----
        -*- Thermal drivers  --->
        [*] Watchdog Timer Support  --->
        < > Sonics Silicon Backplane support  ----
        < > Broadcom specific AMBA  ----
            Multifunction device drivers  --->
        -*- Voltage and Current Regulator Support  --->
        < > Remote Controller support  ----
        [ ] HDMI CEC drivers  ----
        < > Multimedia support  ----
            Graphics support  --->
        <*> Sound card support  --->
            HID support  --->
        [*] USB support  --->
        < > MMC/SD/SDIO card support  ----
        < > Sony MemoryStick card support  ----
        -*- LED Support  --->
        [ ] Accessibility support  ----
        < > InfiniBand support  ----
        [*] Real Time Clock  --->
        [*] DMA Engine support  --->
            DMABUF options  --->
        [ ] Auxiliary Display support  ----
        < > Userspace I/O drivers  ----
        < > VFIO Non-Privileged userspace driver framework  ----
        ┴(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

进入“Backlight & LCD device support”，并选中以下选项

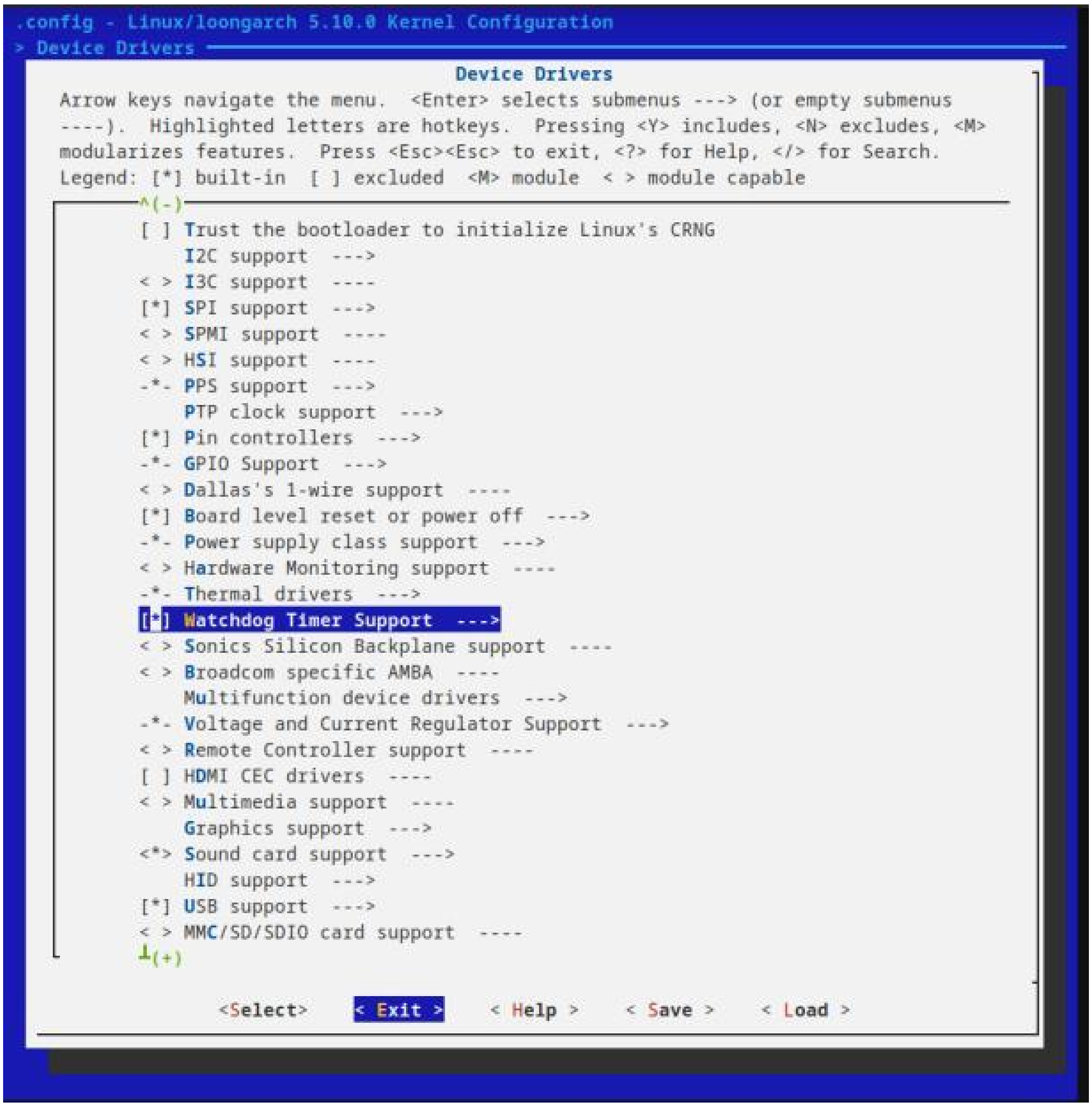


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Graphics support ─────────────────────────────────────
                        Graphics support
   Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
   ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
   modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
   Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ┌───────^(-)──────────────────────────────────────────────────────────────
  │       [ ]   Enable pci device driver support for DC in LS7A Bridge
  │       <M> Kernel modesetting driver for loongson display controller (NEW)
  │       [*]   Enable pci device driver support for DC in LS7A1000 Bridge (NEW)
  │       < > Matrox G200
  │       < > R-Car Gen3 and RZ/G2 DU HDMI Encoder Support
  │       < > R-Car DU LVDS Encoder Support
  │       < > QXL virtual GPU
  │       < > DRM Support for bochs dispi vga interface (qemu stdvga)
  │           Display Panels  --->
  │           Display Interface Bridges  --->
  │       < > ETNAVIV (DRM support for Vivante GPU IP cores)
  │       < > ARC PGU
  │       < > i.MX (e)LCDIF LCD controller
  │       < > Cirrus driver for QEMU emulated device
  │       < > GM12U320 driver for USB projectors
  │       < > DRM support for HX8357D display panels
  │       < > DRM support for ILI9225 display panels
  │       < > DRM support for ILI9341 display panels
  │       < > DRM support for ILI9486 display panels
  │       < > DRM support for MI0283QT
  │       < > DRM support for Pervasive Displays RePaper panels (V231)
  │       < > DRM support for Sitronix ST7586 display panels
  │       < > DRM support for Sitronix ST7715R/ST7735R display panels
  │       [ ] Enable legacy drivers (DANGEROUS)  ----
  │           Frame buffer Devices  --->
  │           Backlight & LCD device support  --->
  │           Console display driver support  --->
  │       [*] Bootup logo  --->
  └

              <Select>    < Exit >    < Help >    < Save >    < Load >
```


```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Graphics support > Backlight & LCD device support
                         Backlight & LCD device support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        < > Lowlevel LCD controls
        -*- Lowlevel Backlight controls
        < >   Backlight Driver for Kinetic KTD253
        <*>   Generic PWM based Backlight Driver
        < >   Qualcomm PMIC WLED Driver
        < >   Backlight Driver for ADP8860/ADP8861/ADP8863 using WLED
        < >   Backlight Driver for ADP8870 using WLED
        < >   Backlight Driver for LM3630A
        < >   Backlight Driver for LM3639
        < >   Backlight driver for TI LP855X
        <*>   Generic GPIO based Backlight Driver
        < >   Sanyo LV5207LP Backlight
        < >   Rohm BD6107 Backlight
        < >   Backlight driver for the Arctic Sands ARCxCnnnn family
        < >   Generic LED based Backlight Driver

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

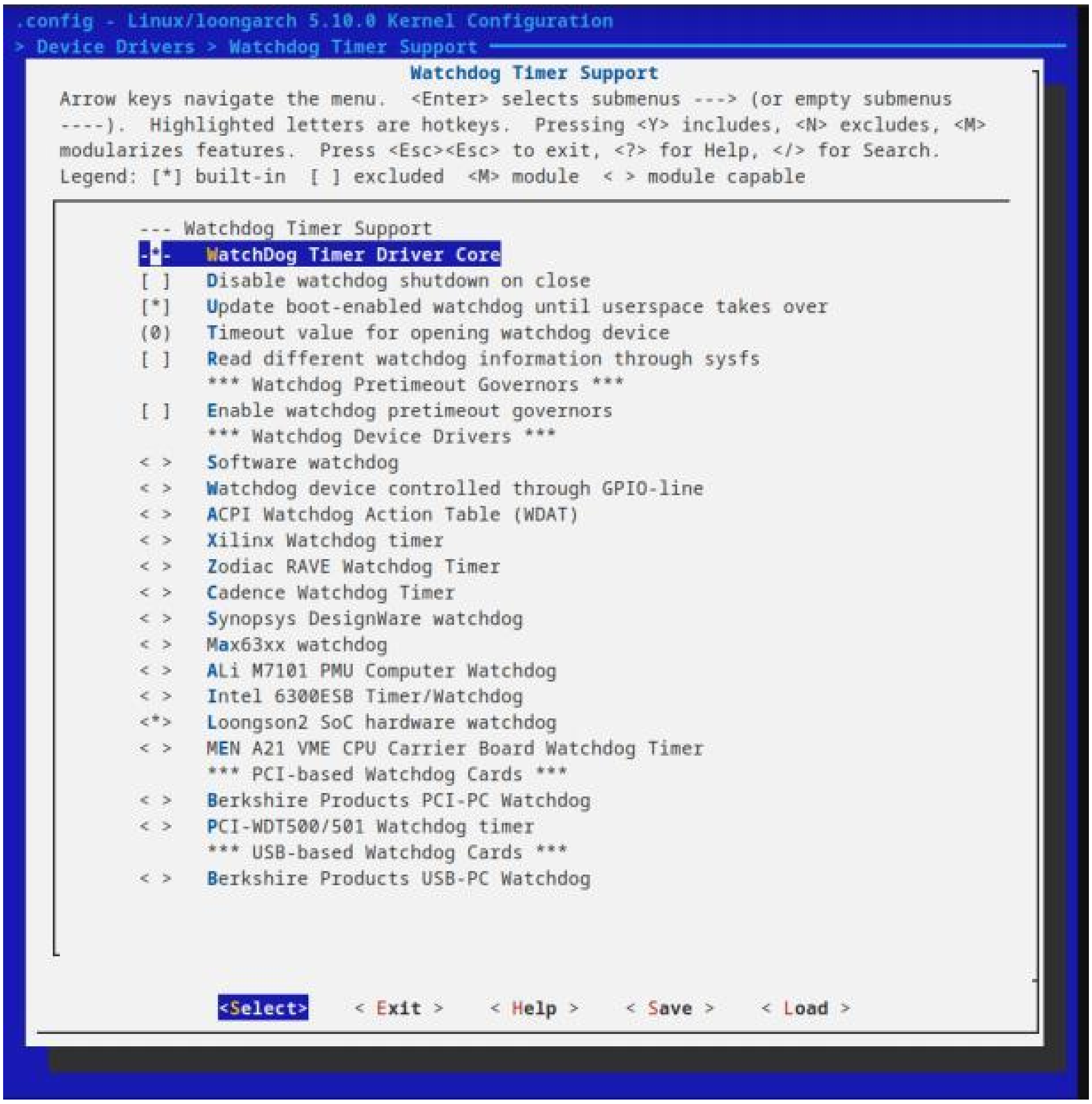
10.2.12.WATCHDOG驱动

Device Drivers 选中”Watchdog Timer Support”选项，按空格选为“*”，按回车进入



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers ─────────────────────────────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        ^(-)
        [ ] Trust the bootloader to initialize Linux's CRNG
            I2C support  --->
        < > I3C support  ----
        [*] SPI support  --->
        < > SPMI support  ----
        < > HSI support  ----
        -*- PPS support  --->
            PTP clock support  --->
        [*] Pin controllers  --->
        -*- GPIO Support  --->
        < > Dallas's 1-wire support  ----
        [*] Board level reset or power off  --->
        -*- Power supply class support  --->
        < > Hardware Monitoring support  ----
        -*- Thermal drivers  --->
        [*] Watchdog Timer Support  --->
        < > Sonics Silicon Backplane support  ----
        < > Broadcom specific AMBA  ----
            Multifunction device drivers  --->
        -*- Voltage and Current Regulator Support  --->
        < > Remote Controller support  ----
        [ ] HDMI CEC drivers  ----
        < > Multimedia support  ----
            Graphics support  --->
        <*> Sound card support  --->
            HID support  --->
        [*] USB support  --->
        < > MMC/SD/SDIO card support  ----
        ┴(+)

           <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中下图中的选项，按空格选为“*”。



```
.config - Linux/loongarch 5.10.0 Kernel Configuration
> Device Drivers > Watchdog Timer Support ─────────────────────────────────
                          Watchdog Timer Support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Watchdog Timer Support
        -*-   WatchDog Timer Driver Core
        [ ]   Disable watchdog shutdown on close
        [*]   Update boot-enabled watchdog until userspace takes over
        (0)   Timeout value for opening watchdog device
        [ ]   Read different watchdog information through sysfs
              *** Watchdog Pretimeout Governors ***
        [ ]   Enable watchdog pretimeout governors
              *** Watchdog Device Drivers ***
        < >   Software watchdog
        < >   Watchdog device controlled through GPIO-line
        < >   ACPI Watchdog Action Table (WDAT)
        < >   Xilinx Watchdog timer
        < >   Zodiac RAVE Watchdog Timer
        < >   Cadence Watchdog Timer
        < >   Synopsys DesignWare watchdog
        < >   Max63xx watchdog
        < >   ALi M7101 PMU Computer Watchdog
        < >   Intel 6300ESB Timer/Watchdog
        <*>   Loongson2 SoC hardware watchdog
        < >   MEN A21 VME CPU Carrier Board Watchdog Timer
              *** PCI-based Watchdog Cards ***
        < >   Berkshire Products PCI-PC Watchdog
        < >   PCI-WDT500/501 Watchdog timer
              *** USB-based Watchdog Cards ***
        < >   Berkshire Products USB-PC Watchdog

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

10.3.驱动列表

代码模块与编译控制宏

驱动	代码模块	编译控制宏	备注
UART	drivers/tty/serial/8250/8250_core.c	CONFIG_SERIAL_8250	启用 8250/16550 串口驱动
DVO	drivers/gpu/drm/loongson/lsdc_platform_drv.c	CONFIG_DRM_LOONGSON	龙芯 drm 驱 动
I2C	drivers/i2c/busses/i2c-ls2x.c	CONFIG_I2C_LS2X	龙芯 I2C 驱 动
NETWORK	drivers/net/ethernet/stmicro/stmmac/dwmac- generic.c	CONFIG_DWMAC_GENERIC	启用 DWMAC 驱 动
GPIO	drivers/gpio/gpio-loongson.c	CONFIG_GPIO_LOONGSON	龙芯 gpio 驱 动

CAN	drivers/net/can/sja1000/sja1000_platform.c	CONFIG_CAN_SJA1000_PLATFORM	系统总线支持SJA1000 驱动
SPI	drivers/spi/spi-ls.c	CONFIG_SPI_LS	龙芯 spi 驱 动
SPI FLASH	drivers/mtd/spi-nor/core.c	CONFIG_MTD_SPI_NOR	支持 spi-nor 设备
NAND	drivers/mtd/nand/raw/ls-nand.c	CONFIG_MTD_NAND_LS	龙芯 nand 驱动
PWM	drivers/pwm/pwm-ls.c	CONFIG_PWM_LS	龙芯 pwm 驱动
RTC	drivers/rtc/rtc-ls2x.c	CONFIG_RTC_DRV_LS2X	龙芯 RTC 驱 动
WATCHDOG	drivers/watchdog/loongson2_wdt.c	CONFIG_LOONGSON2_WDT	龙芯2K watch dog 驱动
LED	drivers/leds/leds-pwm.c	CONFIG_LEDS_PWM	基于 pwm 的 led 驱动
	drivers/leds/leds-gpio.c	CONFIG_LEDS_GPIO	基于 gpio 的 led 驱动
KEYS	drivers/input/keyboard/gpio_keys_polled.c	CONFIG_KEYBOARD_GPIO_POLLED	轮询式 gpio 按键驱动
eMMC/SDIO	drivers/mmc/host/ls2kmci.c	CONFIG_MMC_LS2K	eMMC/SDIO 驱动

10.4.设备树选择

先锋派的40PIN支持多种复用配置（复用配置关系见第2章板卡简介的引脚复用图），默认的配置是**功能一复用**，另外提供 额外的两个设备树名为**ls2k300_pai_func2.dts**、 **ls2k300_pai_pihat.dts**。分别是“**功能二复用**”（包括**CAN**、 **UART4-5- 6-9**），和**树莓派HAT**（包括**温度传感器**、 **OLED屏幕**）的设备树。

使用时将设备树覆盖原有的 **ls2k300_pai.dts**，再重新编译，将编译后的文件更新到板卡即可。下面以使用**功能二复用为 例**。

```
$ cp arch/loongarch/boot/dts/loongson/ls2k300_pai_func2.dts

arch/loongarch/boot/dts/loongson/ls2k300_pai.dts
编译新的内核
$ make uImage

或只编译dtb，在arch/loongarch/boot/dts/loongson 目录下会生成 ls2k300_pai.dtb $ ma
ke dtbs
```

十一、 U-Boot 定制与裁减

11.0 U-boot 特性

基于 U-boot-2022.04版本移植开发

- 支持USB1.0（ OHCI ），USB2.0(EHCI)，
暂不支持otg • 支持双网口
- 支持SATA
- 显示最大分辨率1920x1080，支持双屏显示，支持
bmp logo • 支持I2C设备
- 支持EMMC
- 支持SPI Flash 用于启动BootLoader
- 支持菜单更新系统，包括更新u-boot、 Linux内核和根文件系统等，可通过网络（tftp）或U盘进行 更新

11.1 如何编译U-Boot

```
设置交叉工具链等环境
$ source ./set_env.sh
====>setup env for LoongArch...

指定板卡配置
$ make loongson_2k300_pai_defconfig
make[1]: Entering directory '/home/vm/u-boot-2022.04'
  GEN     Makefile
  HOSTCC  scripts/kconfig/conf.o
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config #
make[1]: Leaving directory '/home/vm/u-boot-2022.04'

开始编译
$ make
make[1]: Entering directory '/home/vm/u-boot-2022.04'
  GEN     Makefile
scripts/kconfig/conf  --syncconfig Kconfig
  CFG     u-boot.cfg
  GEN     include/autoconf.mk.dep
  CFG     spl/u-boot.cfg
  GEN     include/autoconf.mk
  CC      spl/lib/elf.o
  AR      spl/lib/built-in.o LD
  spl/u-boot-spl
  OBJCOPY spl/u-boot-spl-nodtb.bin
  SYM     spl/u-boot-spl.sym
  CAT     spl/u-boot-spl-dtb.bin
  COPY    spl/u-boot-spl.bin
  CAT     u-boot-with-spl.bin
make[1]: Leaving directory '/home/vm/u-boot-2022.04'
```

编译成功之后会生成 `u-boot-with-spl.bin`

11.2 如何修改内核引导参数

```
$ make menuconfig
Boot options >
    [*] Enable boot arguments
    (console=ttyS0,115200 rw noinitrd init=/sbin/init rootfstype=ext4 rootwait) Boot
```

或者直接修改 defconfig文件中CONFIG_BOOTARGS中的值

```
$ grep "CONFIG_BOOTARGS" configs/loongson_2k300_pai_defconfig -n

32:CONFIG_BOOTARGS="console=ttyS0,115200 rw noinitrd init=/sbin/init
rootfstype=ext4 rootwait"
```

11.3 如何修改logo

`U-boot` 的 `logo` 文件是在 `./tools/Makefile`中控制的，通过`Makefile` 可知，可以在编译时设置 `LOGO_BMP` 的值来指定使用具体的 `logo`文件。如果没有指定则优先查找 `./tools/logos/` 目录下有没 有当前配置的 `$BOARD.bmp`， 之后是 `$VENDOR.bmp`，再最后就是 `U-boot` 的默认 `logo` 了。

```
# Generic logo
ifeq ($(LOGO_BMP),)
LOGO_BMP= $(srctree)/$(src)/logos/denx.bmp

# Use board logo and fallback to vendor
ifneq ($(wildcard $(srctree)/$(src)/logos/$(BOARD).bmp),)
LOGO_BMP= $(srctree)/$(src)/logos/$(BOARD).bmp
else
ifneq ($(wildcard $(srctree)/$(src)/logos/$(VENDOR).bmp),)
LOGO_BMP= $(srctree)/$(src)/logos/$(VENDOR).bmp
endif
```

```
endif

endif #  !LOGO_BMP
```

在`config.mk`中定义了`BOARD`和 `VENDOR`， `BORAD` 的值为`CONFIG_SYS_BOARD`，`VENDOR` 的值为 `CONFIG_SYS_VENDOR`

```
ARCH  := $(CONFIG_SYS_ARCH:"%"=%)
CPU  := $(CONFIG_SYS_CPU:"%"=%)
ifdef CONFIG_SPL_BUILD
ifdef CONFIG_ARCH_TEGRA
CPU  :=
arm720t endif
endif
BOARD  :=
$(CONFIG_SYS_BOARD:"%"=%) ifneq
($(CONFIG_SYS_VENDOR),)
```

在 `board/loongson/Kconfig`中定义了 `CONFIG_SYS_BOARD`和 `CONFIG_SYS_VENDOR`的值

```
config SYS_VENDOR
      default "loongson"


if SOC_LS2k300
config SYS_BOARD
      default "ls2k300"
```

修改logo 有两种方法， 一是编译时指定logo，另一种是直接替换原有logo

11.3.1 编译时指定logo

在 `uboot` 源码进行编译时通过 `LOGO_BMP`指定 `logo`文件路径即可。在 `uboot` 源码的 `./tools/logos/` 中有一个 `loongson_logo.bmp`文件，这是用于显示的 `logo`。 打开 `shell`， 对 `uboot` 源码进行编译。

```
CC_PREFIX=/opt/loongson-gnu-toolchain-x86_64-loongarch64-linux-gnu
export PATH=$CC_PREFIX/bin:$PATH
export LD_LIBRARY_PATH=$CC_PREFIX/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$CC_PREFIX/loongarch64-linux-gnu/lib64:$LD_LIBRARY_PATH


export ARCH=loongarch
export CROSS_COMPILE=loongarch64-linux-gnu-
make loongson_2k300_pai_defconfig
make LOGO_BMP=tools/logos/loongson_logo.bmp –j4
```

11.3.2 替换默认的 logo

在 `U-boot`源码中的 `./tools/logos/`文件夹中的 `denx.bmp` 文件是默认的 `logo` 文件，所以可以直接 把需要显示的 `logo` 替换这个 `denx.bmp` 文件，然后重新编译即可。

Logo 文件的要求：格式为bmp 文件， 文件大小不要超过 60KB ，8位色深

11.4 如何修改CPU频率与内存频率

在uboot 源码的 `include/configs/loongson_2k300.h`中定义了， cpu 与内存的频率。基中 `CORE_FREQ` 为CPU 频率， `DDR_FREQ`为内存频率。修改相应的值之后重新编译烧录uboot 即可。

代码定义片段如下：

```
/* Loongson LS2k300 clock configuration. */
#define REF_FREQ              120     //参考时钟固定为120MHz
#define CORE_FREQ              CONFIG_CPU_FREQ //CPU的时钟在  make menuconfig 里面
选择
#define DDR_FREQ                    800      //MEM 800Mhz
#define APB_FREQ                    200     //SB 100~200MHz, for BOOT, USB, APB, SDIO
#define NET_FREQ                    200      //NETWORK 200~400MHz, for NETWORK, DC
```

11.5 如何在屏幕与串口同步显示调试信息

`U-boot` 中控制输出重定向的变量是 `stdout` 和 `stderr`。在 `U-boot` 命令行模式下，键入 `printenv` 可查看当前的变量值。

如果只想串口显示信息那么可以键入命令：

```
=> setenv stdout serial   （设置只是串口输出）
=> saveenv              (保存到spi flash 中为了重启后依然生效)
```

同理，如果只想屏幕显示信息可以键入命令：

```
=> setenv stdout vidconsole0,vidconsole1 =>

saveenv
```

恢复串口和屏幕同时显示信息可以键入命令：

```
=> setenv stdout serial,vidconsole0,vidconsole1 =>

saveenv
```

`U-boot` 源码修改

针对要烧录的开发板类型，如先锋板，可以修改 `include/configs/loongson_common.h`文件中的 `CONSOLE_STDOUT_SETTINGS`变量。

代码片断如下：

```
#ifdef CONFIG_VIDEO

#define CONSOLE_STDOUT_SETTINGS

    \ "stdin=serial,usbkbd\0" \

    "stdout=serial\0" \

    "stderr=serial,vga\0"

#elif defined(CONFIG_DM_VIDEO)

#define CONSOLE_STDOUT_SETTINGS \

    "splashimage=" __stringify(CONFIG_SYS_LOAD_ADDR) "\0" \

    "stdin=serial,usbkbd\0" \

    "stdout=serial\0" \

    "stderr=serial,vidconsole,vidconsole1\0" #else

#define CONSOLE_STDOUT_SETTINGS \
#endif
```

11.6 U-boot的驱动列表

代码模块与编译控制宏

驱动	代码模块	编译控制宏	备注
Serial	drivers/serial/serial-uclass. c	CONFIG_DM_SERIAL	serial driver model
	drivers/serial/ns16550.c	CONFIG_SYS_NS16550_SERIAL	支持NS16550 UART
GPIO	drivers/gpio/gpio-uclass.c	CONFIG_DM_GPIO	gpio driver model
	drivers/gpio/ls_gpio.c	CONFIG_LOONGSON_GPIO	龙芯GPIO驱动
	cmd/gpio.c	CONFIG_CMD_GPIO	启用uboot gpio指令
GPIO LED	drivers/led/led-uclass.c		led driver model
	cmd/led.c	CONFIG_CMD_LED	启用uboot gpio指令
SPI	drivers/spi/spi-uclass.c	CONFIG_DM_SPI	spi driver model
	drivers/spi/spi-mem.c	CONFIG_SPI_MEM	启用SPI Memory Extension

```
#endif
```

	drivers/spi/ls_spi.c	CONFIG_LOONGSON_SPI	启用龙芯 SPI 驱动

SPI FLASH	drivers/mtd/spi/sf-uclass.c	CONFIG_DM_SPI_FLASH	spi flash driver model
	drivers/mtd/spi/sf_mtd.c	CONFIG_SPI_FLASH_MTD	支持 SPI Flash MTD
	drivers/mtd/spi/spi-nor-core.c	CONFIG_SPI_FLASH	启用spi flash 功能
	drivers/mtd/spi/sf_probe.c		
	drivers/mtd/spi/spi-nor-ids.c		
	cmd/sf.c	CONFIG_CMD_SF	启用 uboot spi flash 指令
MTD	drivers/mtd/mtdcore.c	CONFIG_MTD	启用 mtd 功能
	drivers/mtd/mtduboot.c		
	drivers/mtd/mtd-uclass.c	CONFIG_DM_MTD	mtd driver model
	drivers/mtd/mtdpart.c	CONFIG_MTD_PARTITIONS	启用mtd 分 区功能
	cmd/mtd.c	CONFIG_CMD_MTD	启用 uboot mtd 命令
	cmd/mtdparts.c	CONFIG_CMD_MTDPARTS	启用 uboot mtdparts 命令

LCD	drivers/video/backlight-uclass.c	CONFIG_BACKLIGHT	启用 panel backlight
	drivers/video/backlight_gpio.c	CONFIG_BACKLIGHT_GPIO	启用基于 gpio的背光 控制
	drivers/video/console_normal.c	CONFIG_CONSOLE_NORMAL	启用文本交互界面
	drivers/video/display-uclass.c	CONFIG_DISPLAY	启用 display
	drivers/video/panel-uclass.c	CONFIG_PANEL	启用 panel
	drivers/video/simple_panel.c	CONFIG_SIMPLE_PANEL	启用 simple panel
	drivers/video/u_boot_logo.c	CONFIG_VIDEO_LOGO	启用显示 uboot logo 功能
	drivers/video/vidconsole- uclass.c	CONFIG_DM_VIDEO	lcd/video driver model
	drivers/video/video-uclass.c		
	drivers/video/video_bmp.c		
	drivers/video/loongson_fb.c	CONFIG_VIDEO_LOONGSON	龙芯显示驱动
NETWORK	drivers/net/eth-phy-uclass.c	CONFIG_DM_ETH_PHY	ethernet driver model
	drivers/net/designware.c	CONFIG_ETH_DESIGNWARE	支持 Synopsys Designware Ethernet MAC

	drivers/net/phy/phy.c	CONFIG_PHYLIB	支持以太网 PHY（physical media interface）
	drivers/net/phy/realtek.c	CONFIG_PHY_REALTEK	支持 Realtek 以 太网 PHY

USB	drivers/usb/host/usb-uclass.c	CONFIG_DM_USB	usb driver model
	drivers/usb/host/ohci-generic.c	CONFIG_USB_OHCI_GENERIC	支持 OHCI
	drivers/usb/host/ohci-hcd.c	CONFIG_USB_OHCI_NEW	支持 new OHCI
	drivers/usb/host/ehci-generic.c	CONFIG_USB_EHCI_GENERIC	支持 EHCI
	drivers/usb/host/ehci-hcd.c	CONFIG_USB_EHCI_HCD	支持 EHCI HCD (USB 2.0)
	common/usb_kbd.c	CONFIG_USB_KEYBOARD	支持 USB 键盘
	cmd/usb.c	CONFIG_CMD_USB	启用 uboot usb 命令
I2C	drivers/i2c/i2c-uclass.c	CONFIG_DM_I2C	i2c driver model
	drivers/i2c/ocores_i2c.c	CONFIG_SYS_I2C_OCORES	支持 ocore i2c 驱动
eMMC/SDIO	drivers/mmc/ls_mmc.c	CONFIG_MMC_LOONGSON	支持 mmc 驱动