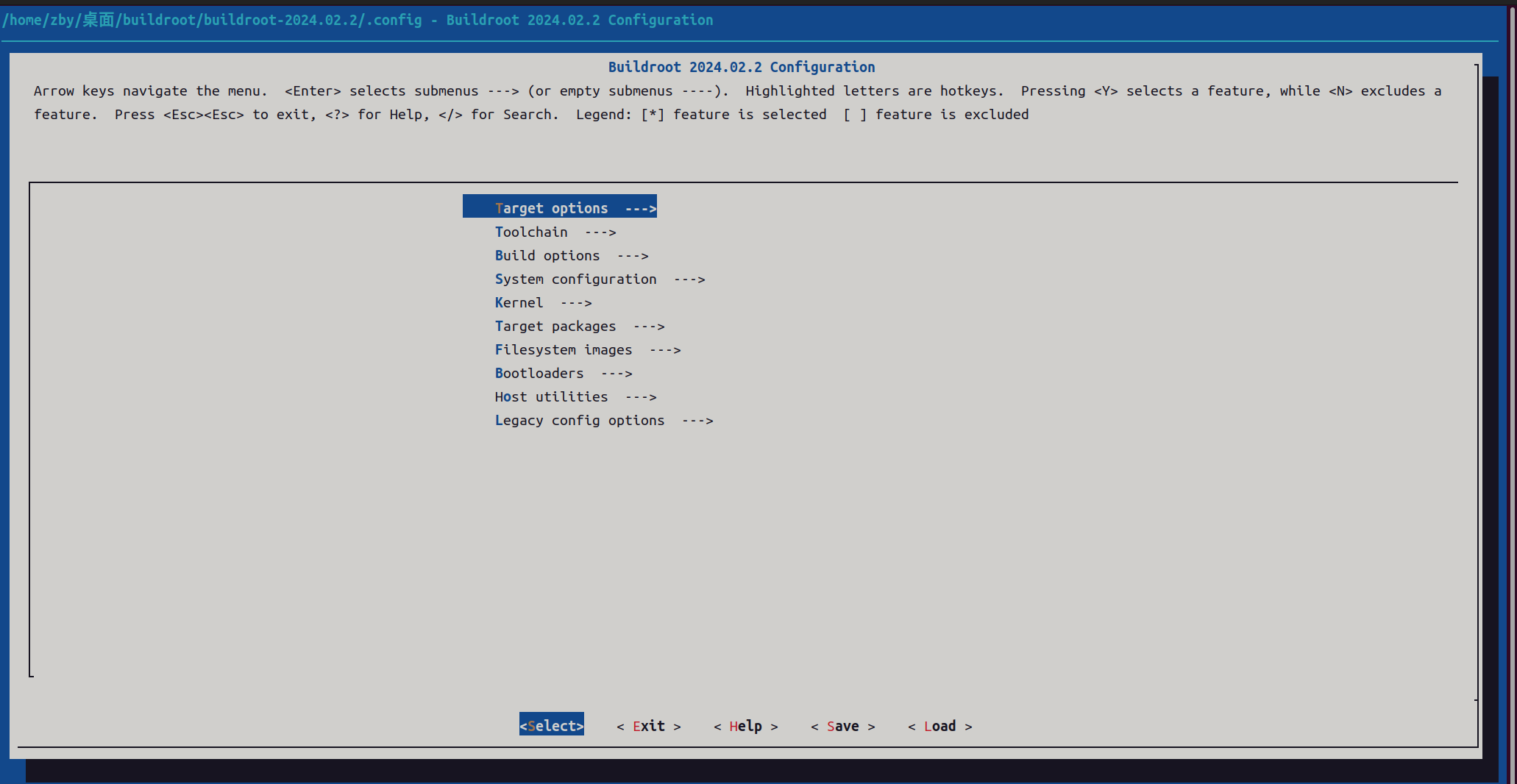
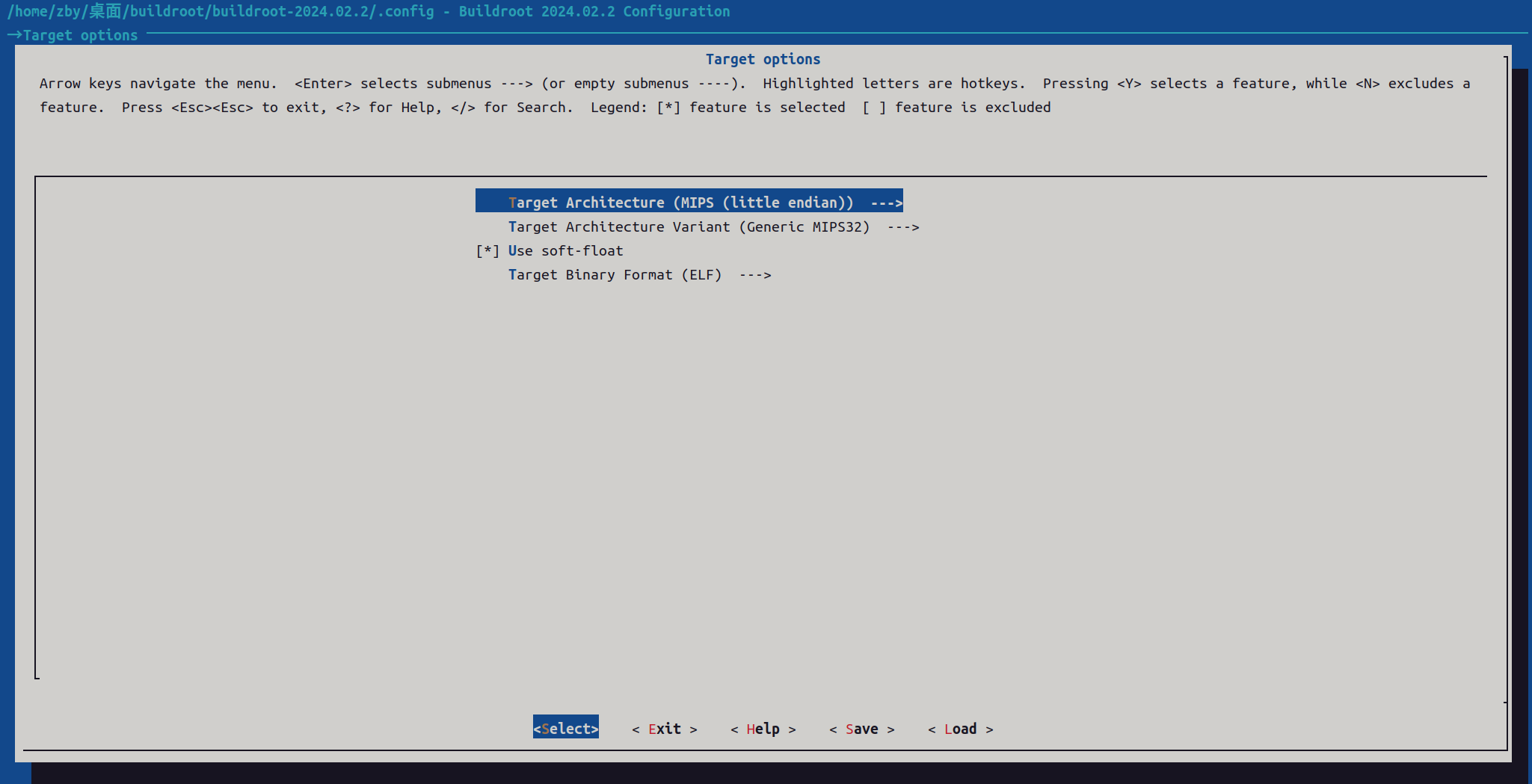
# 1、完成交叉编译工具链生成(生成MIPS大小端工具链) - 截图运行结果

## 1.1小端

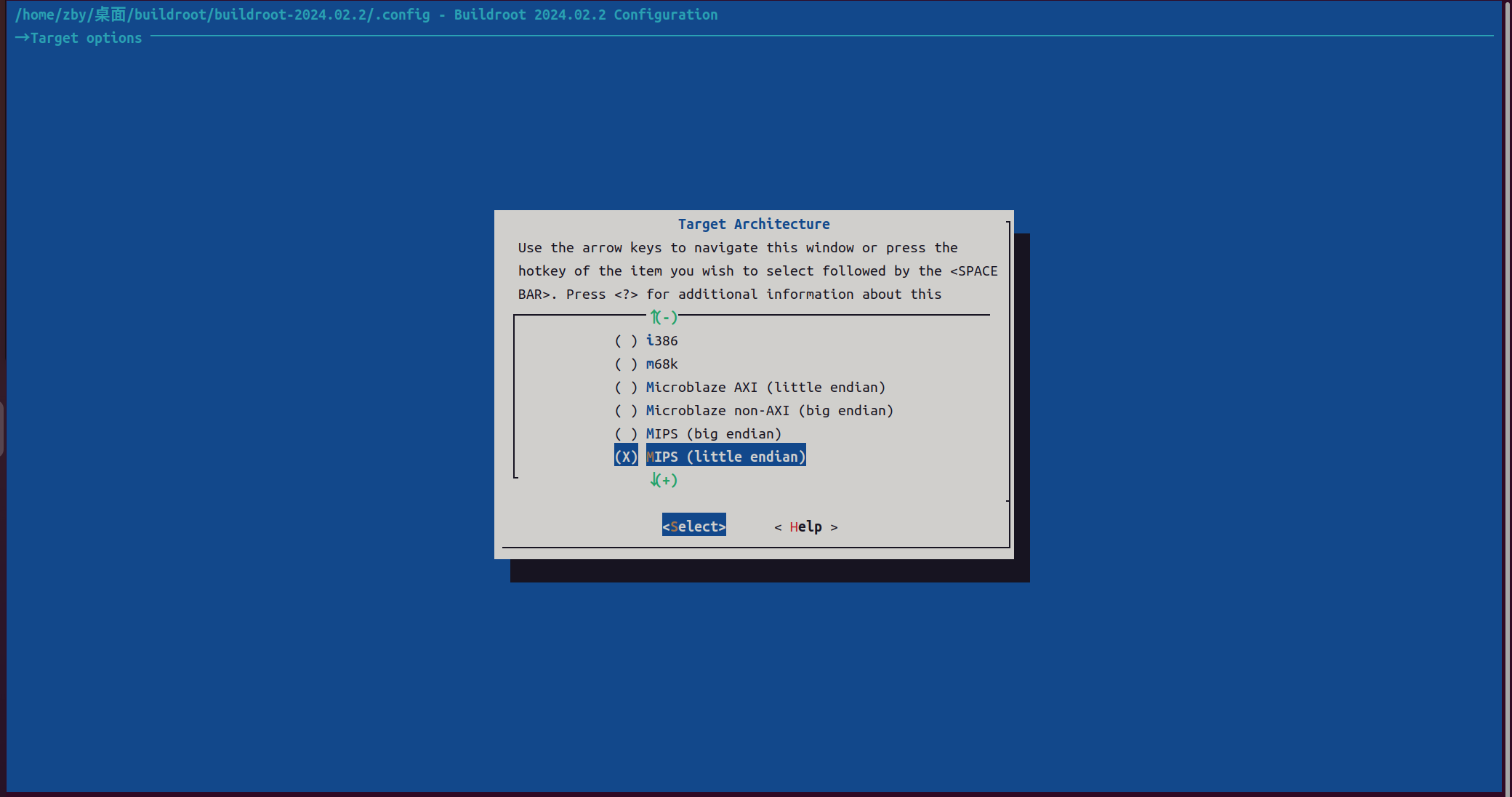
首先在终端中执行指令make menuconfig打开操作界面



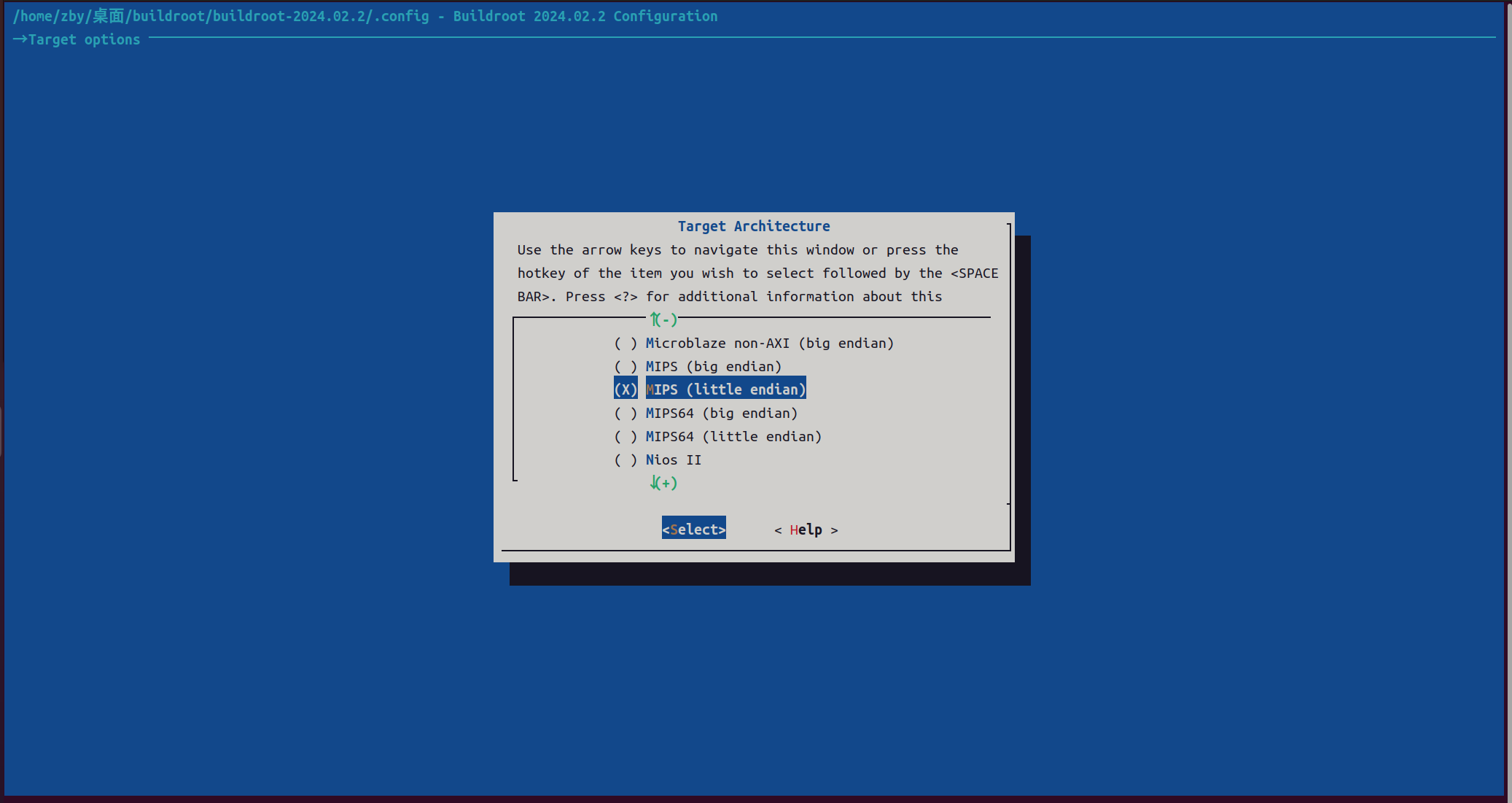
选择Target options



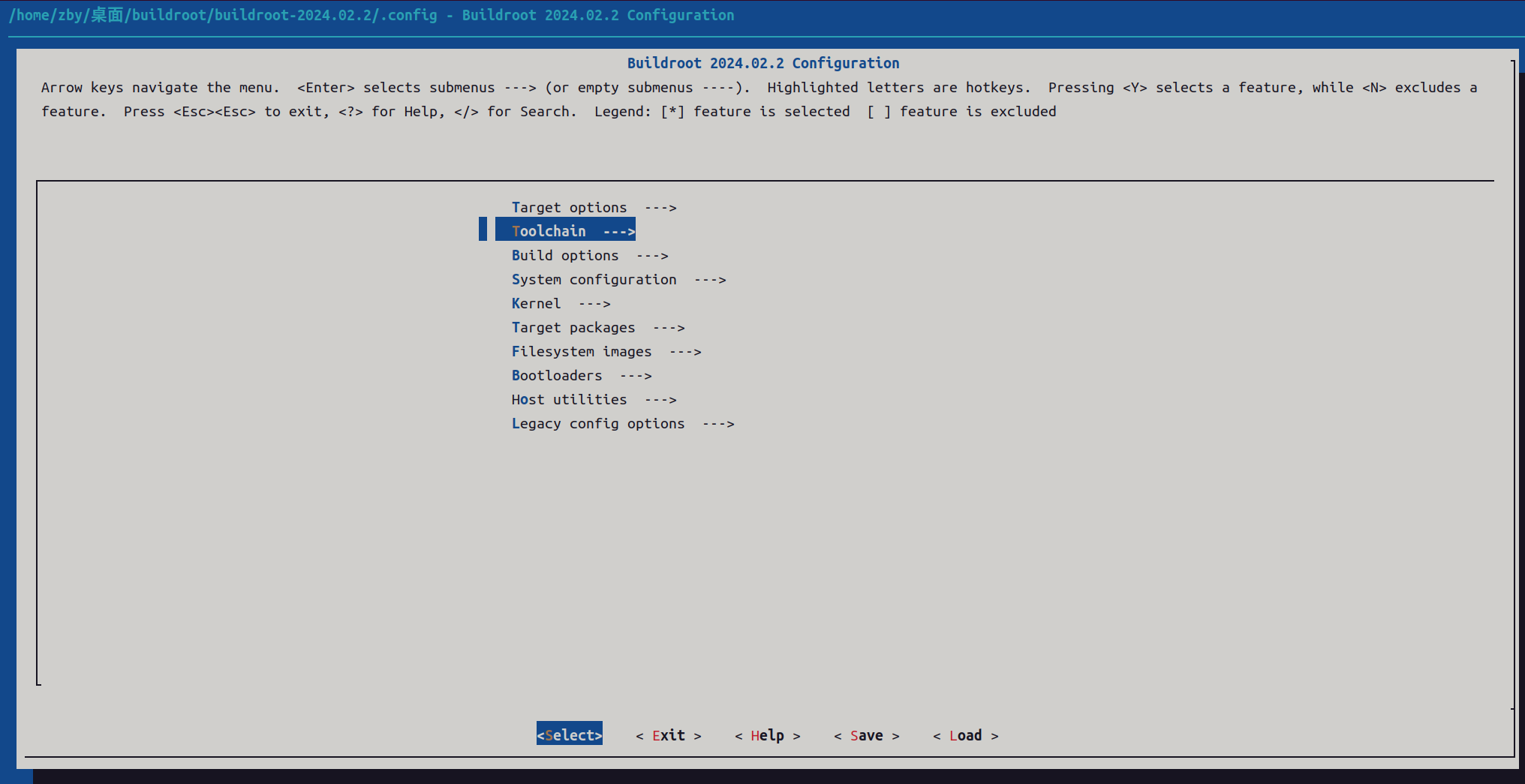
选择更改目标架构



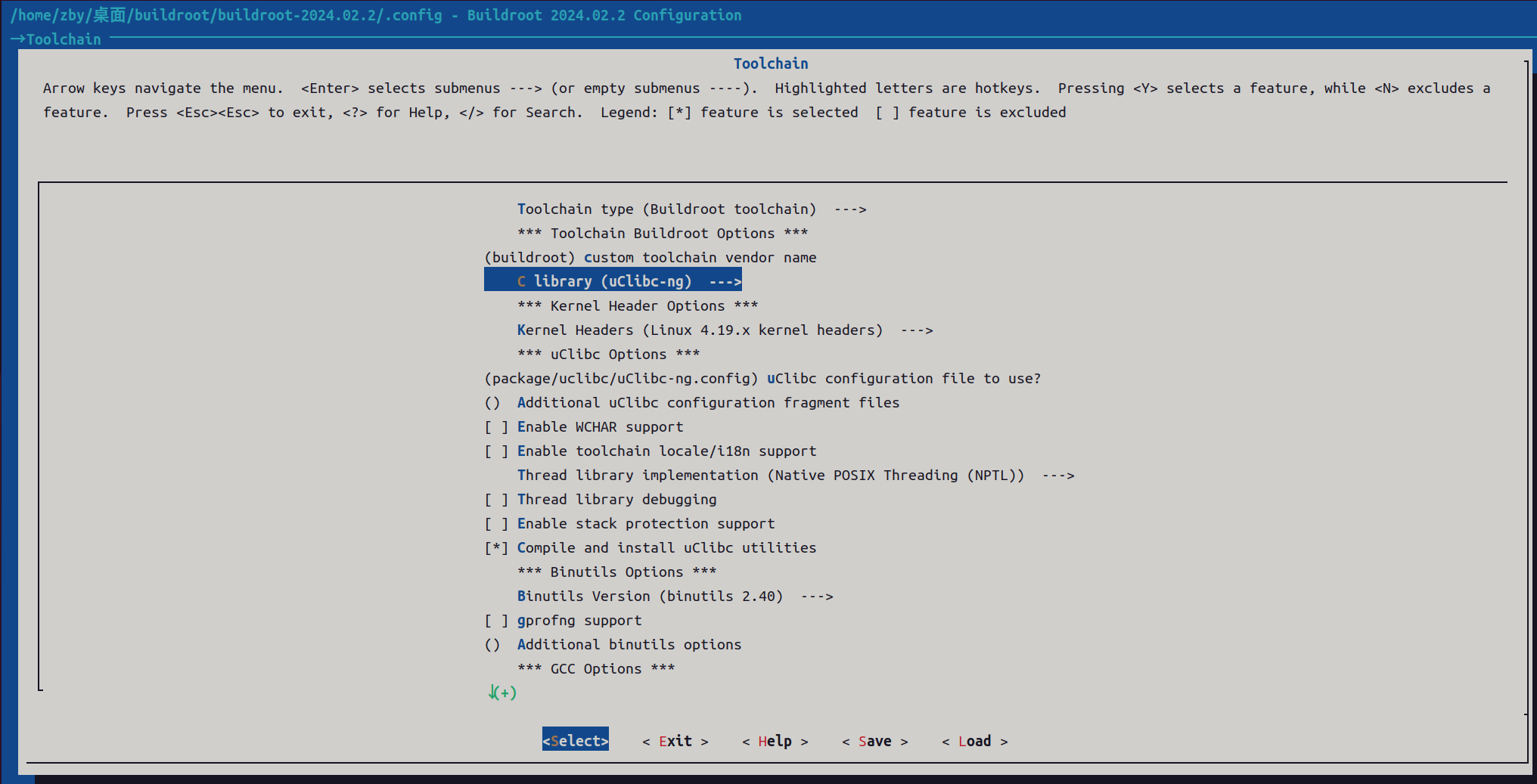
接下来选择MIPS32的小端架构



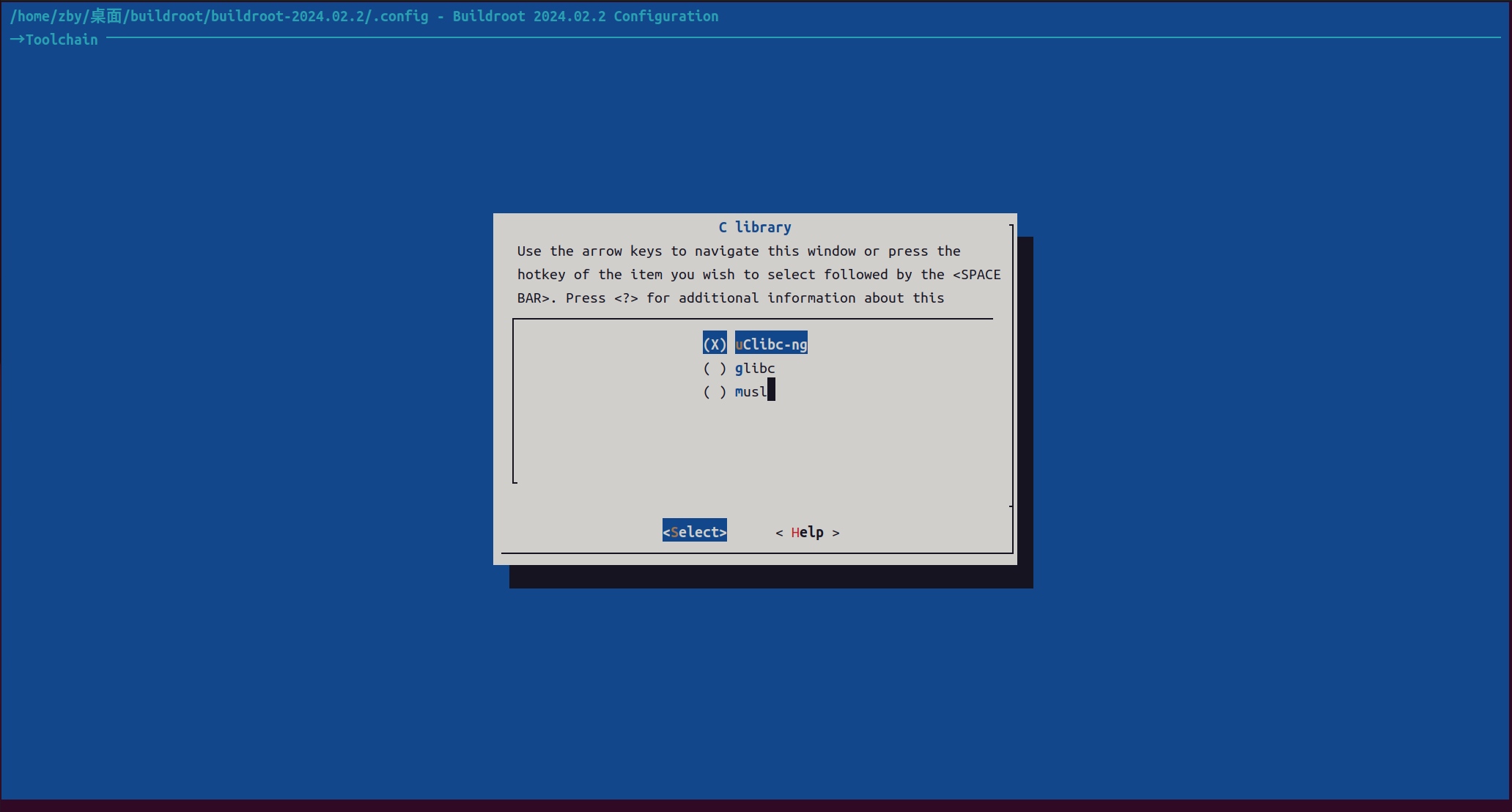
之后退出到根目录界面中选择工具链



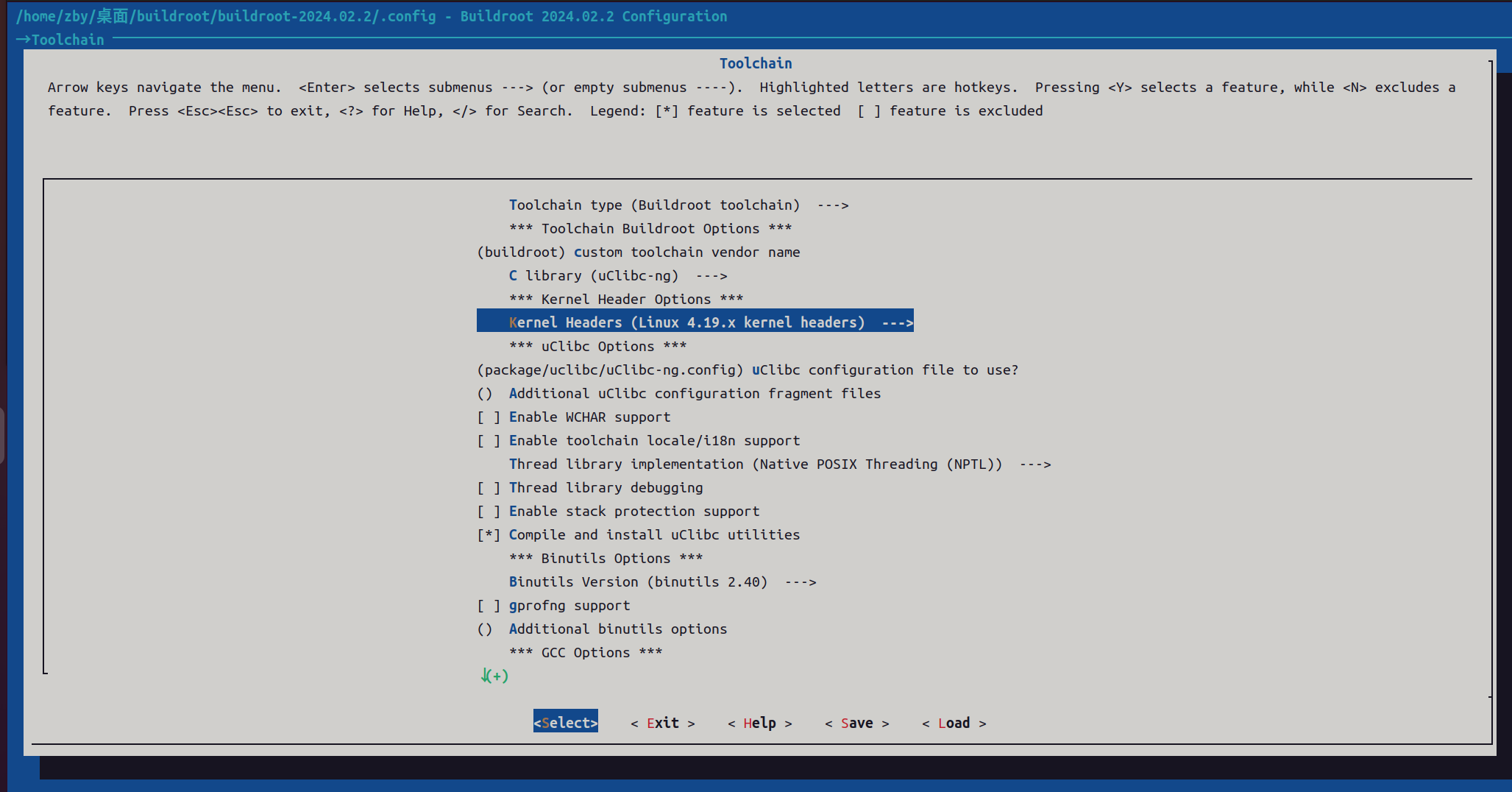
之后选择C语言链接库



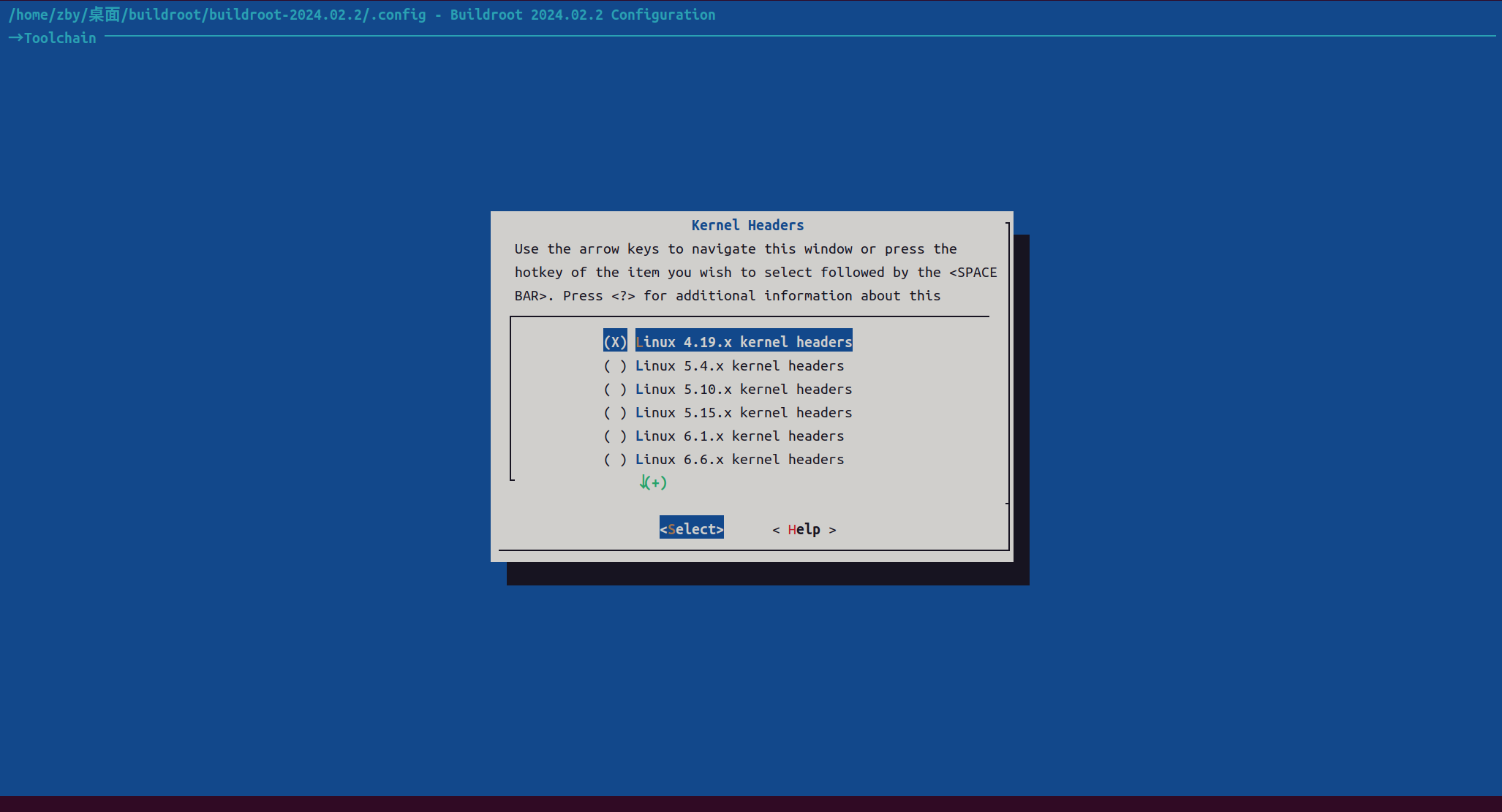
将C语言链接库修改为uClibc-ng



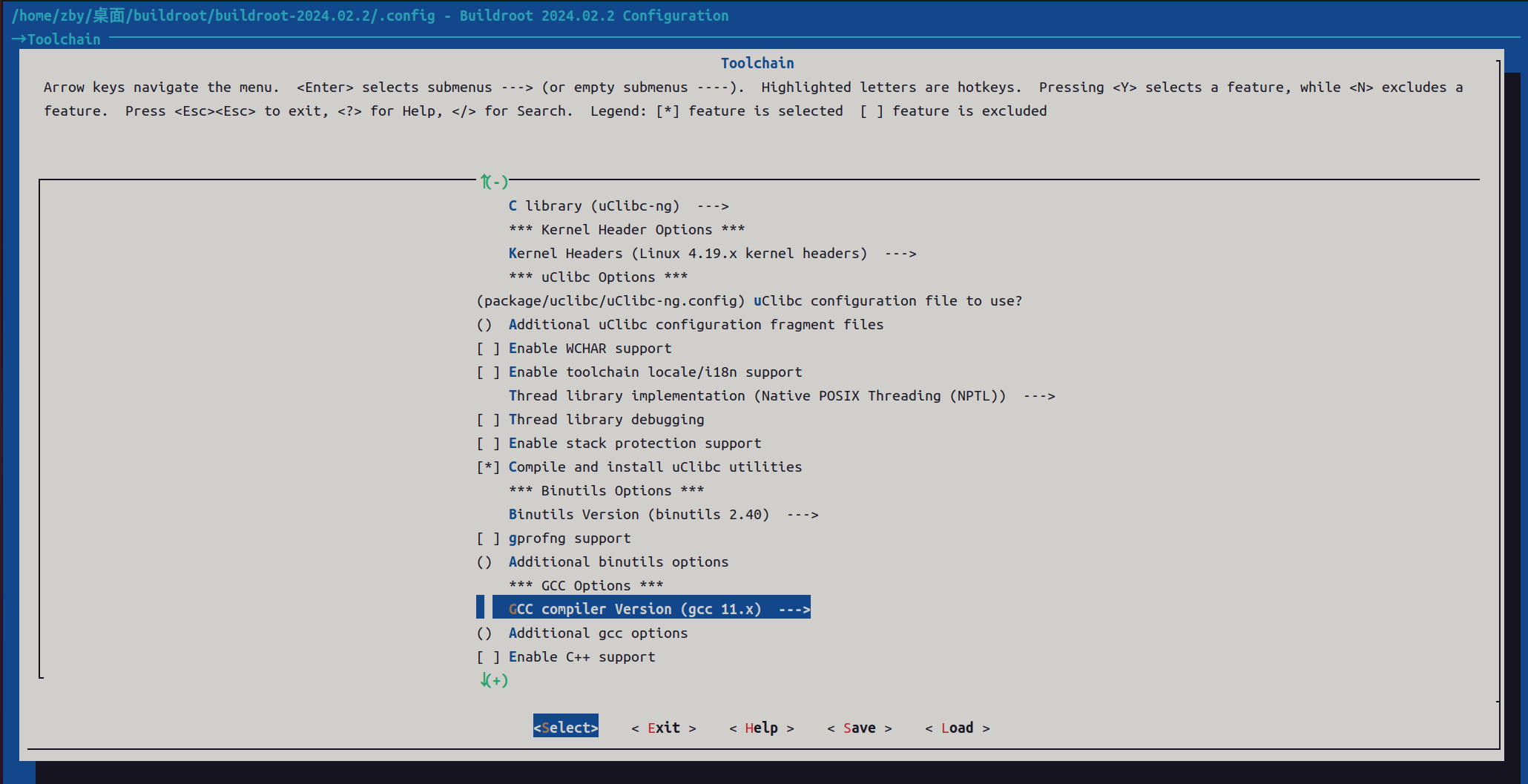
然后选择内核



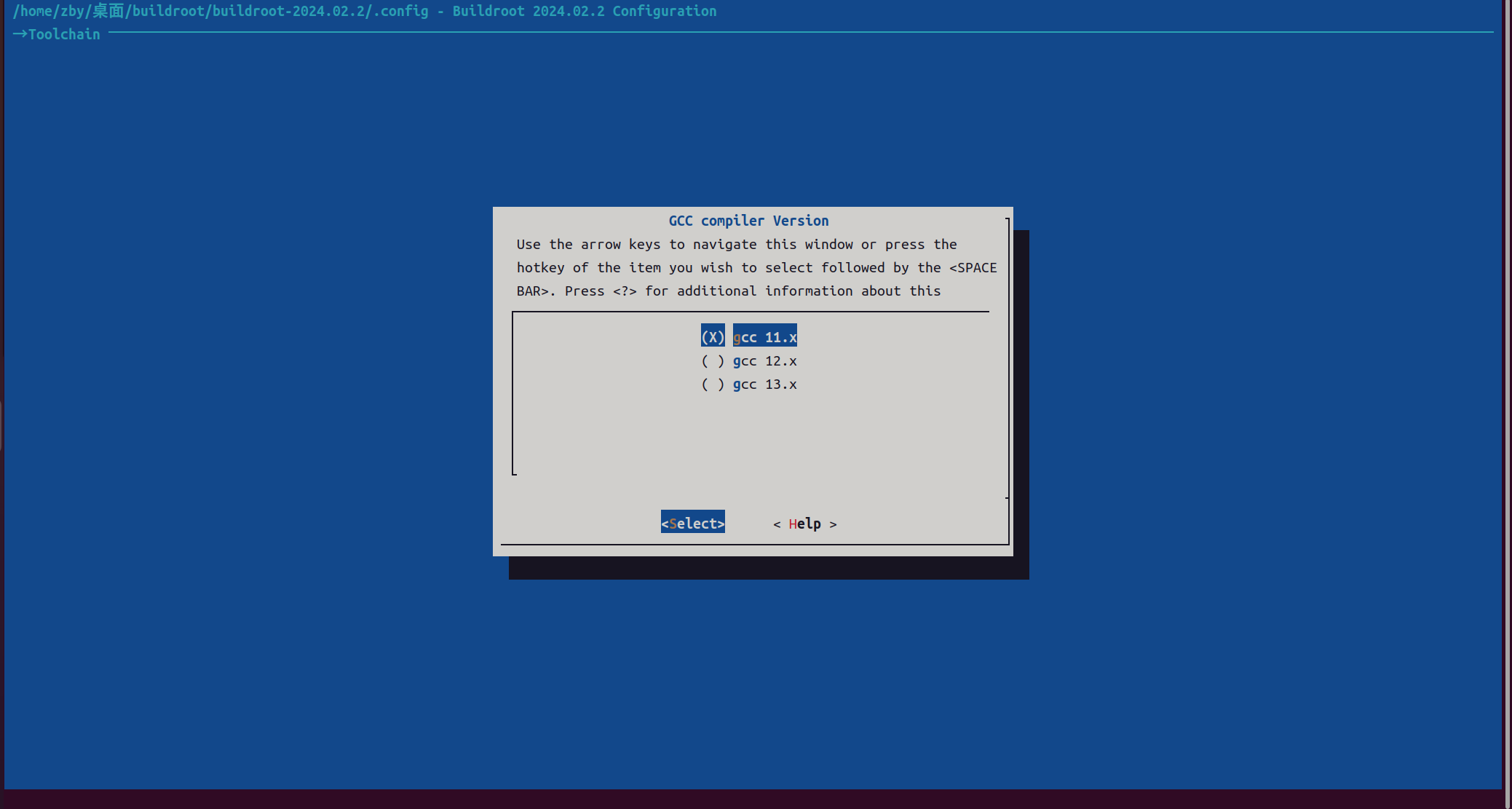
将内核版本选择为Linux 4.19



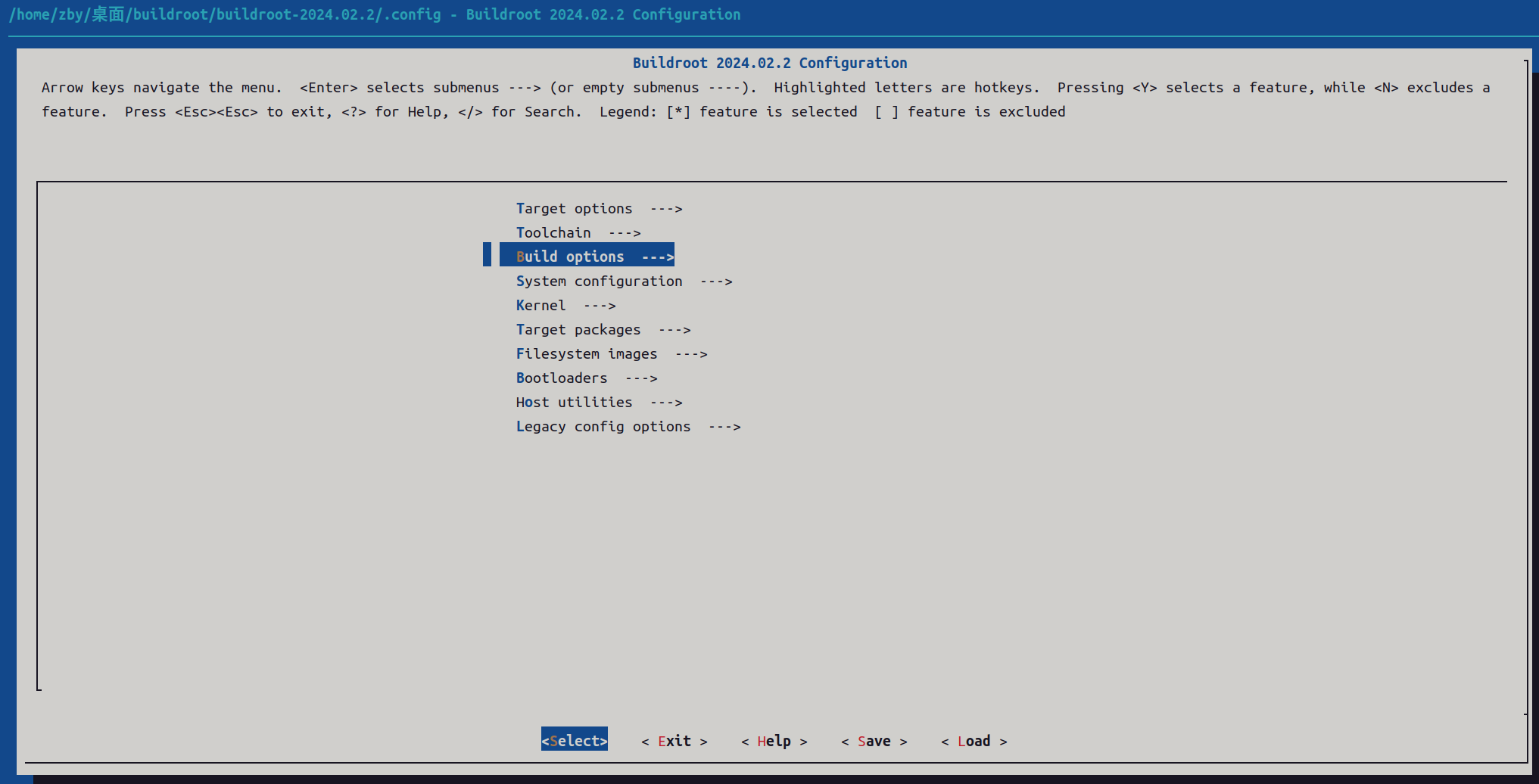
然后选择gcc编译器版本



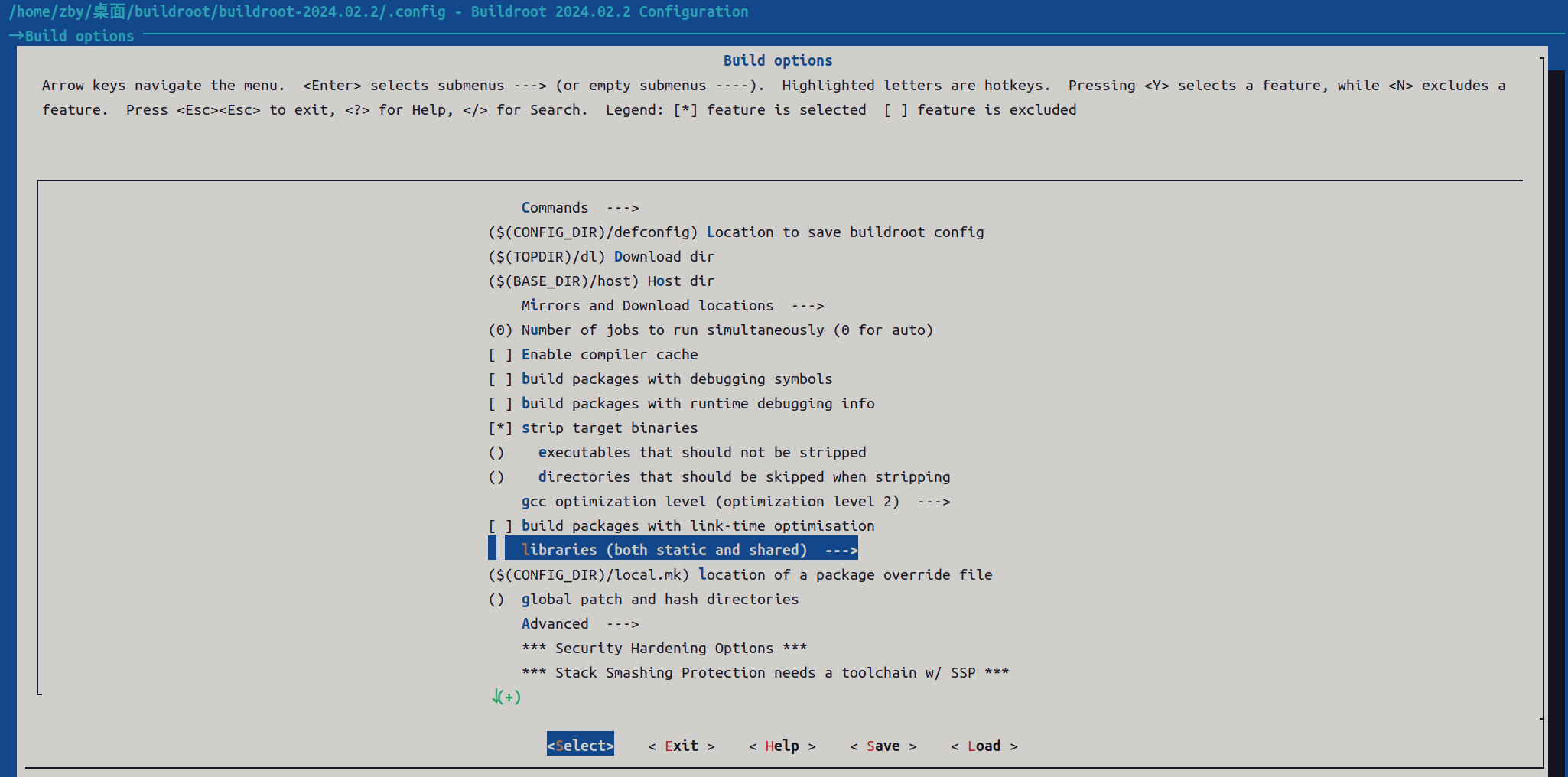
将编译器版本设置为gcc 11.x



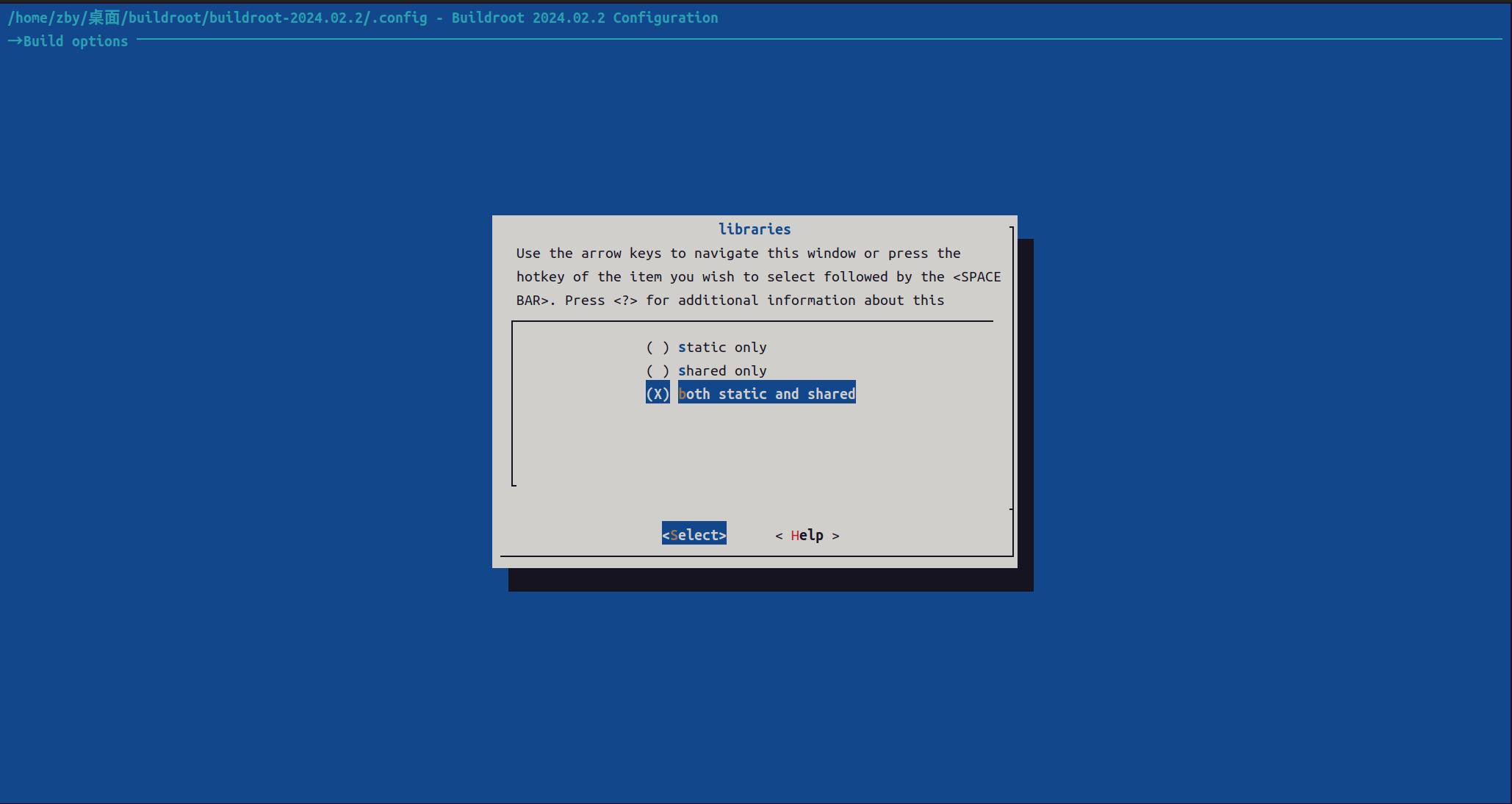
然后回退到主界面选择Build options



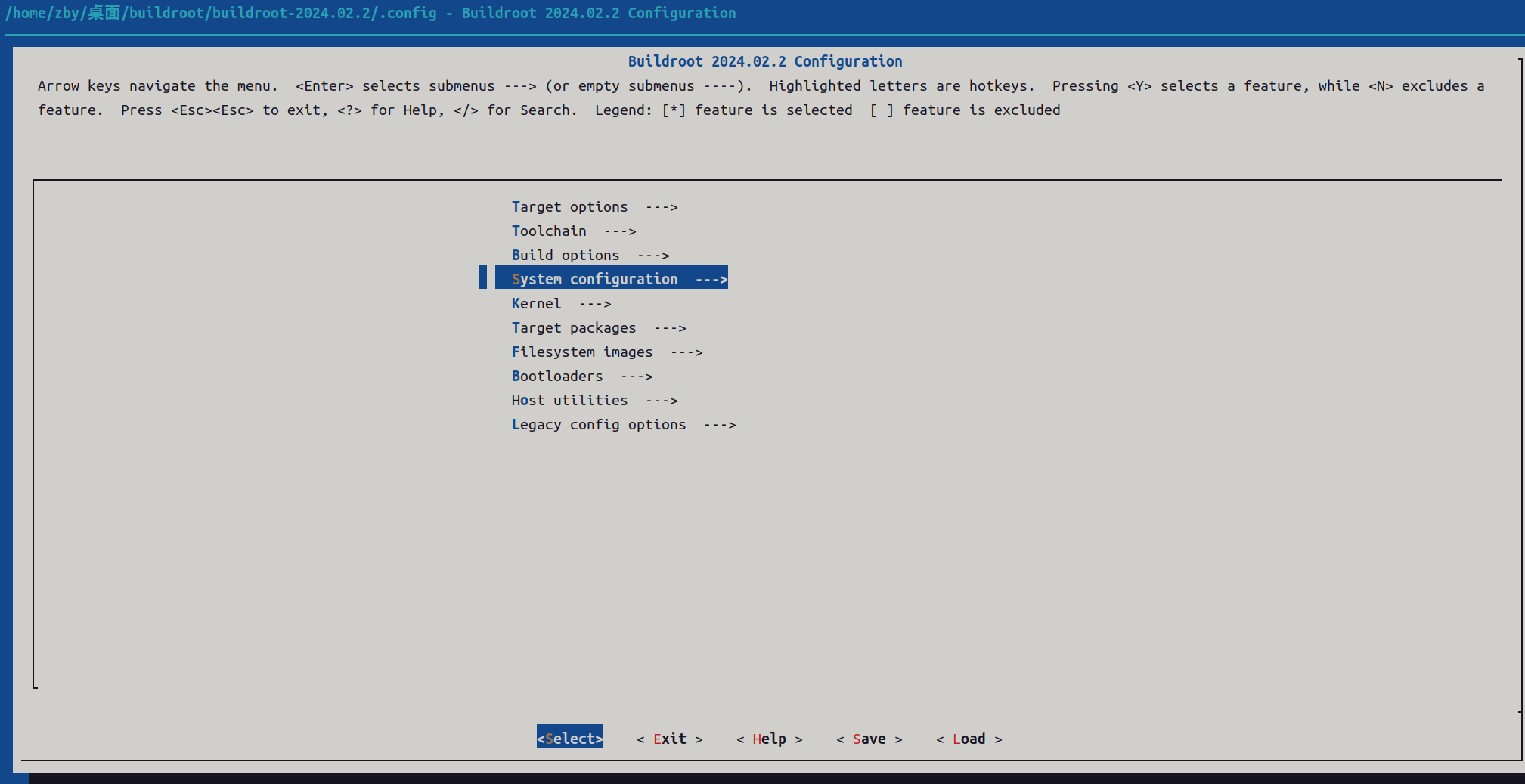
进入libraries选项



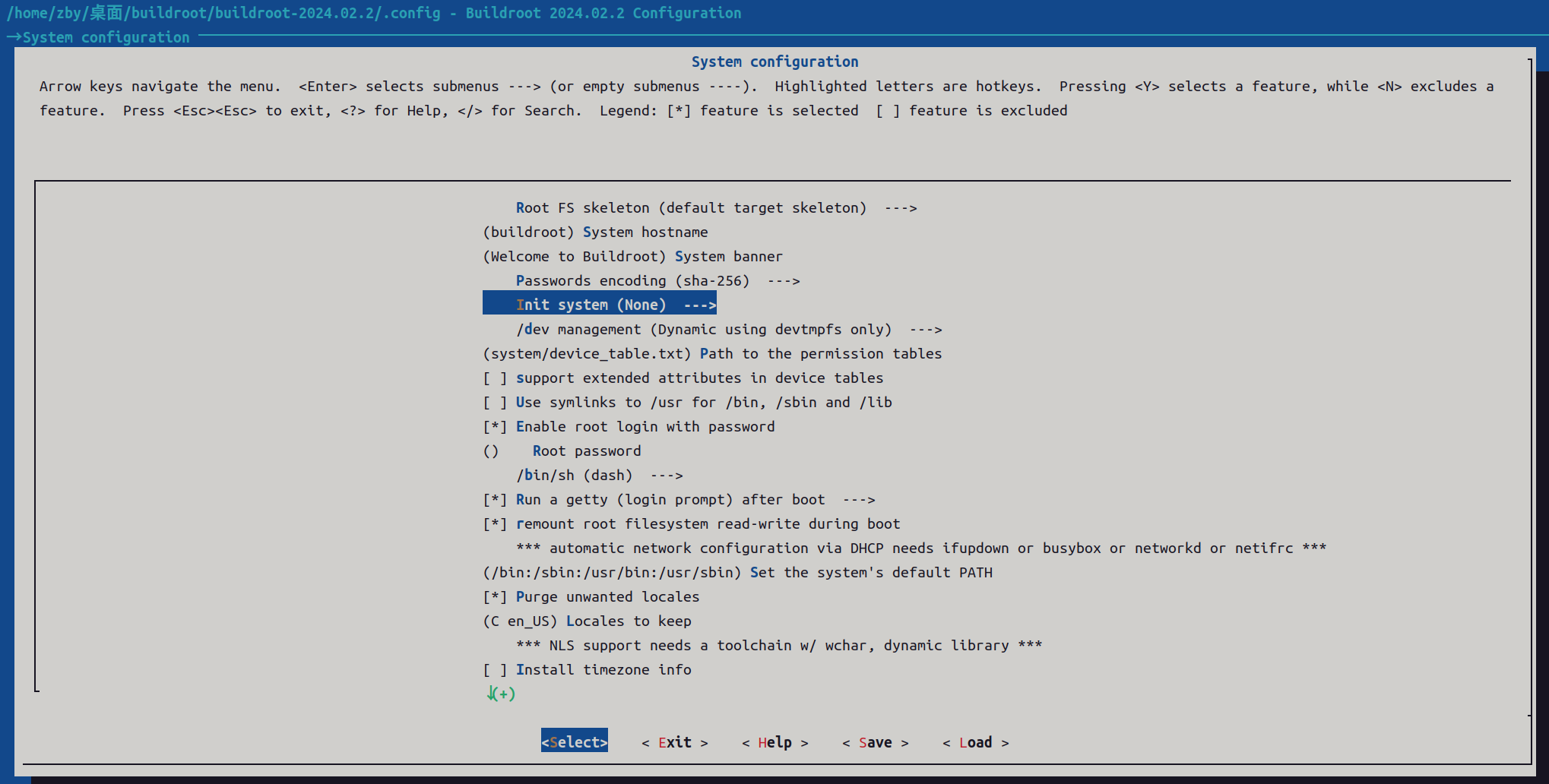
选择支持静态和动态



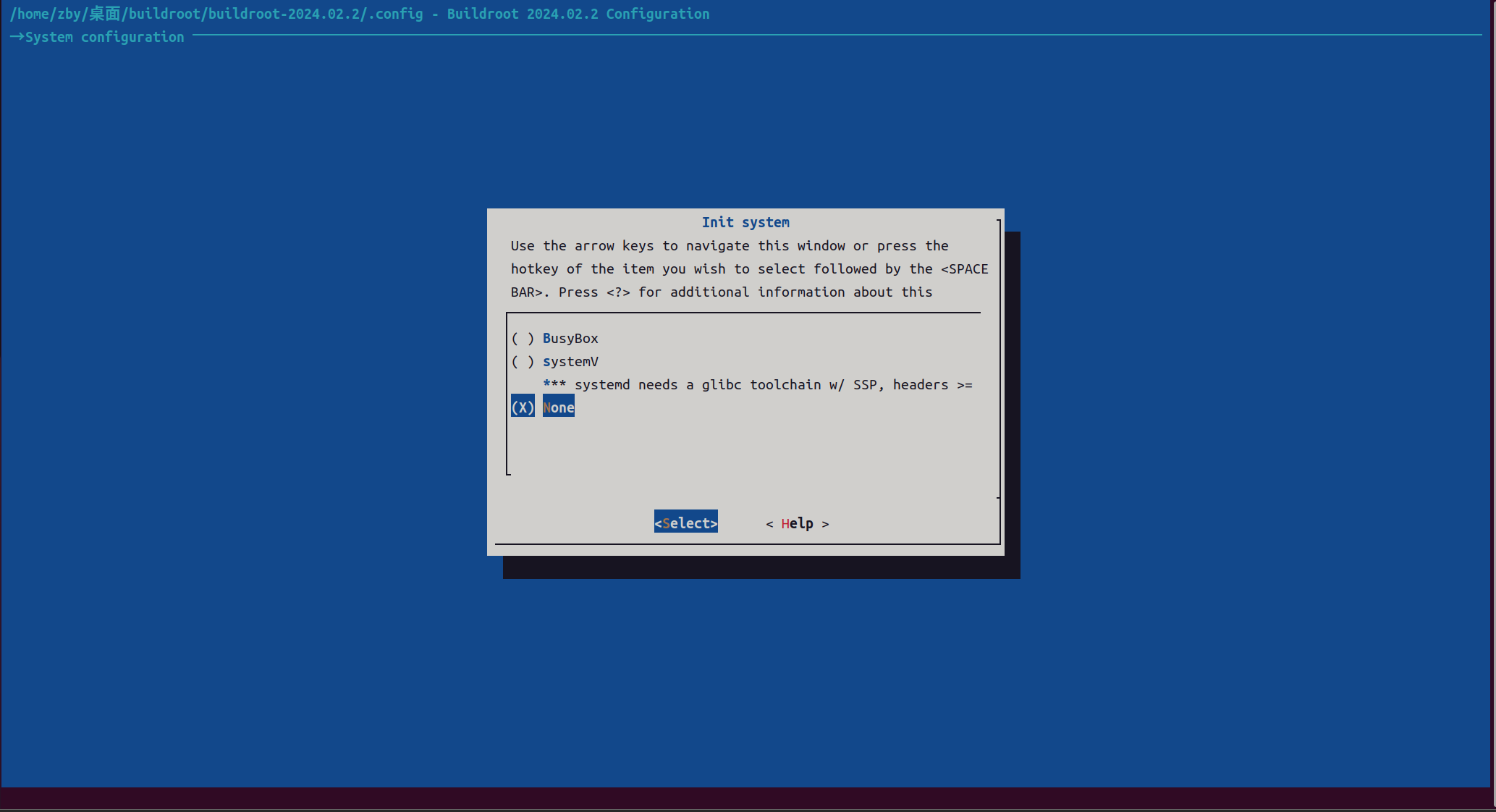
然后回退到主界面进入到系统配置中



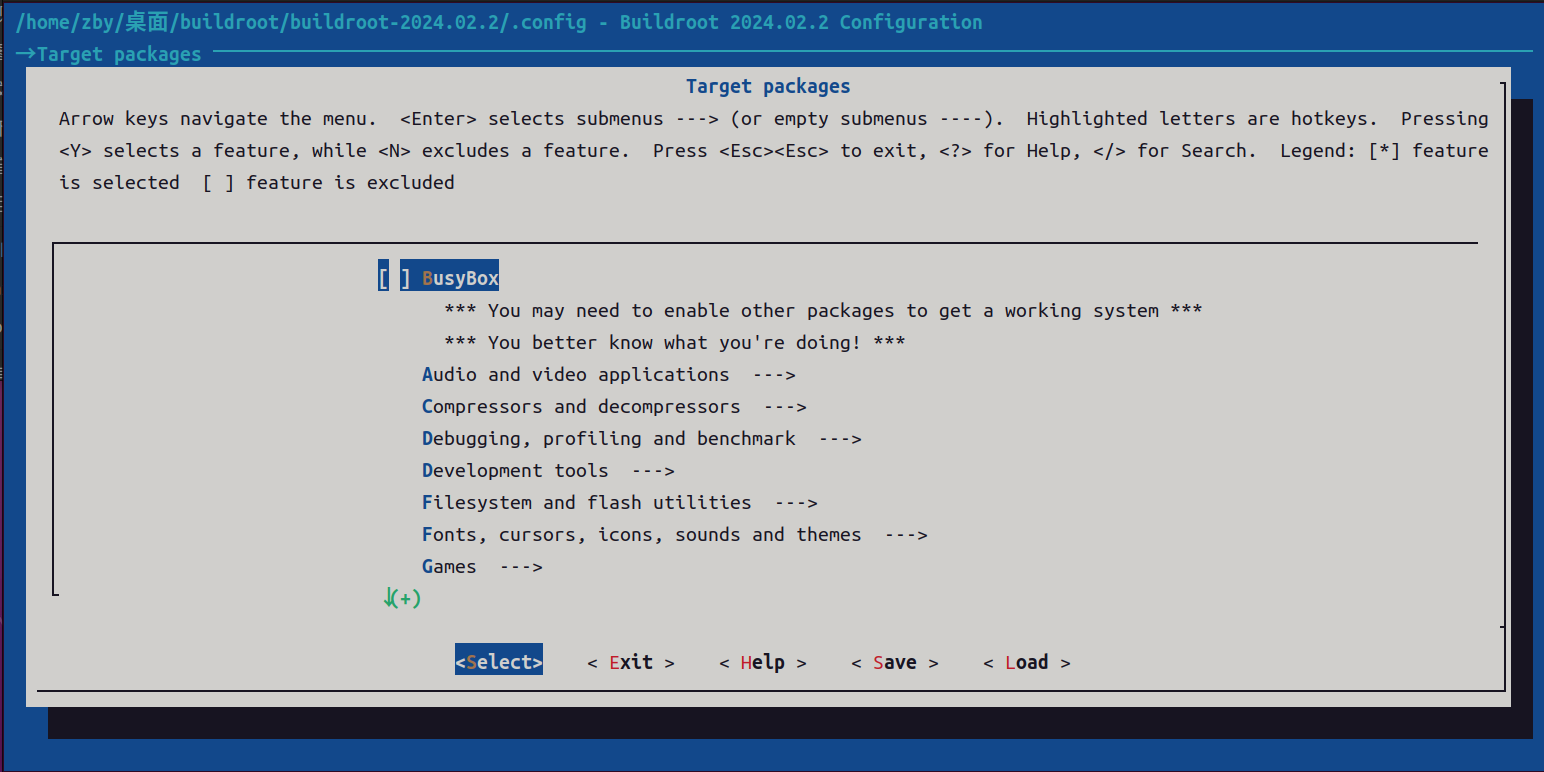
选择Init system选项



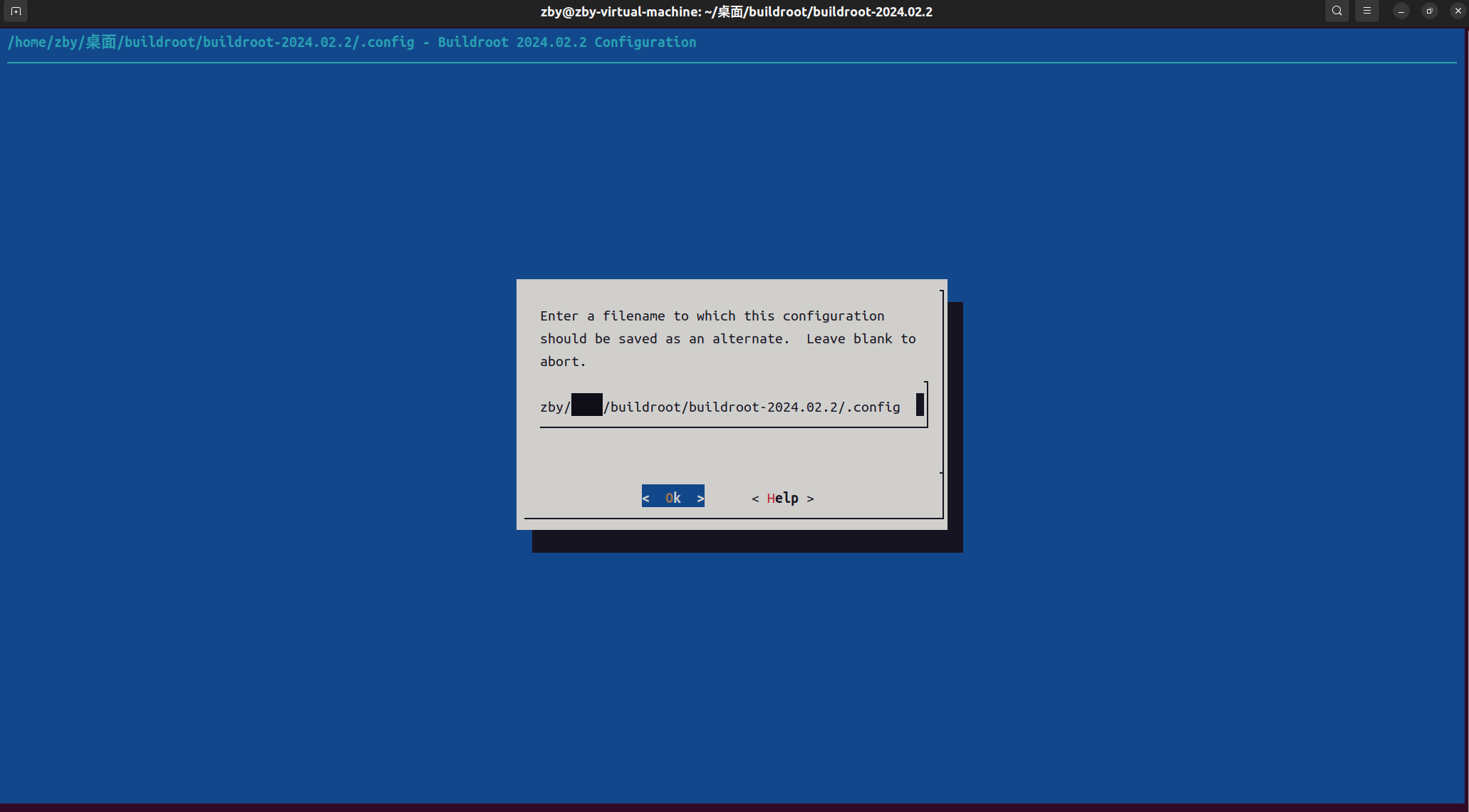
将系统设置为None



进入到Target packages中去掉BusyBox



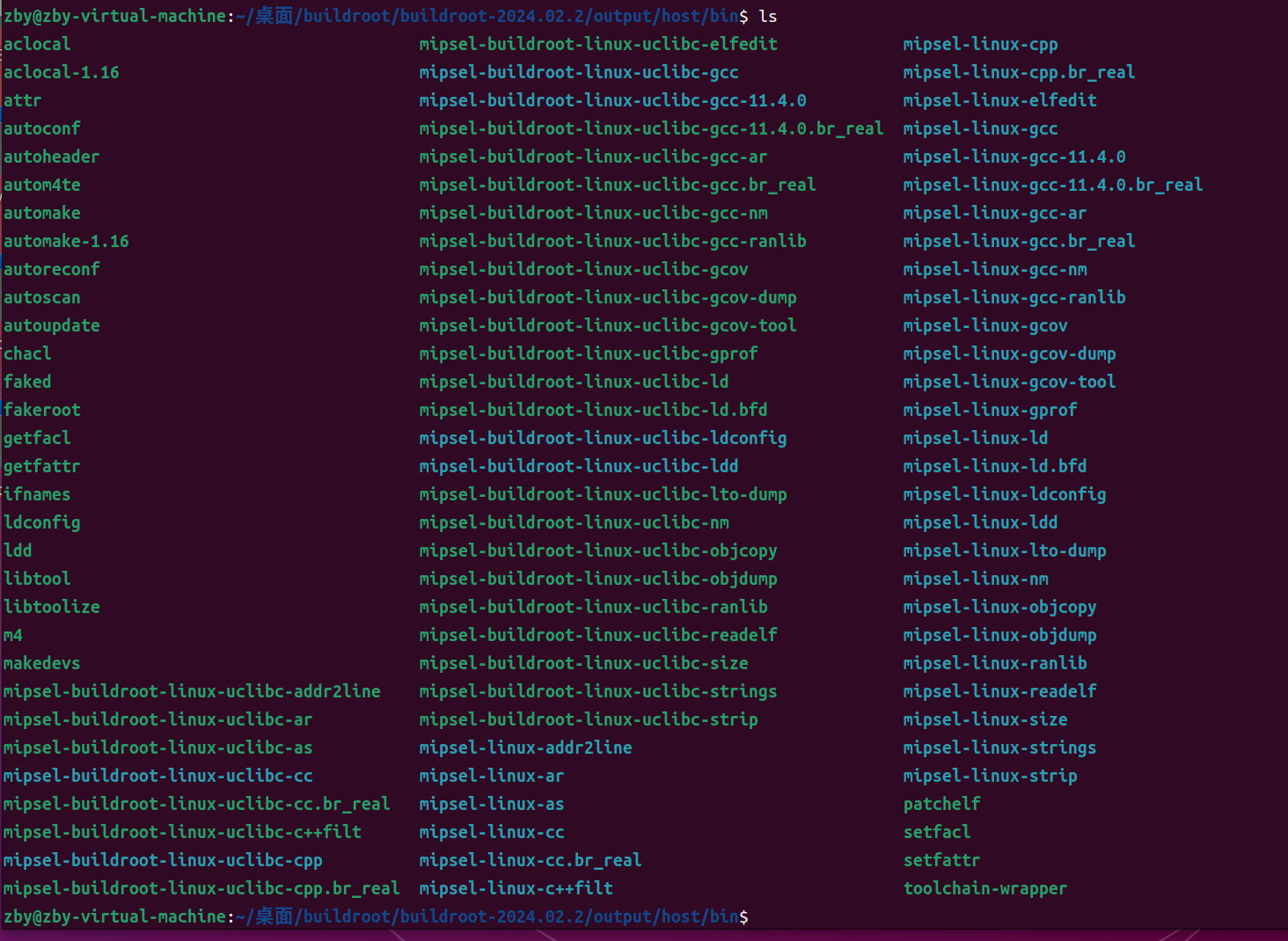
全部设置完成之后返回到主界面，将配置进行保存



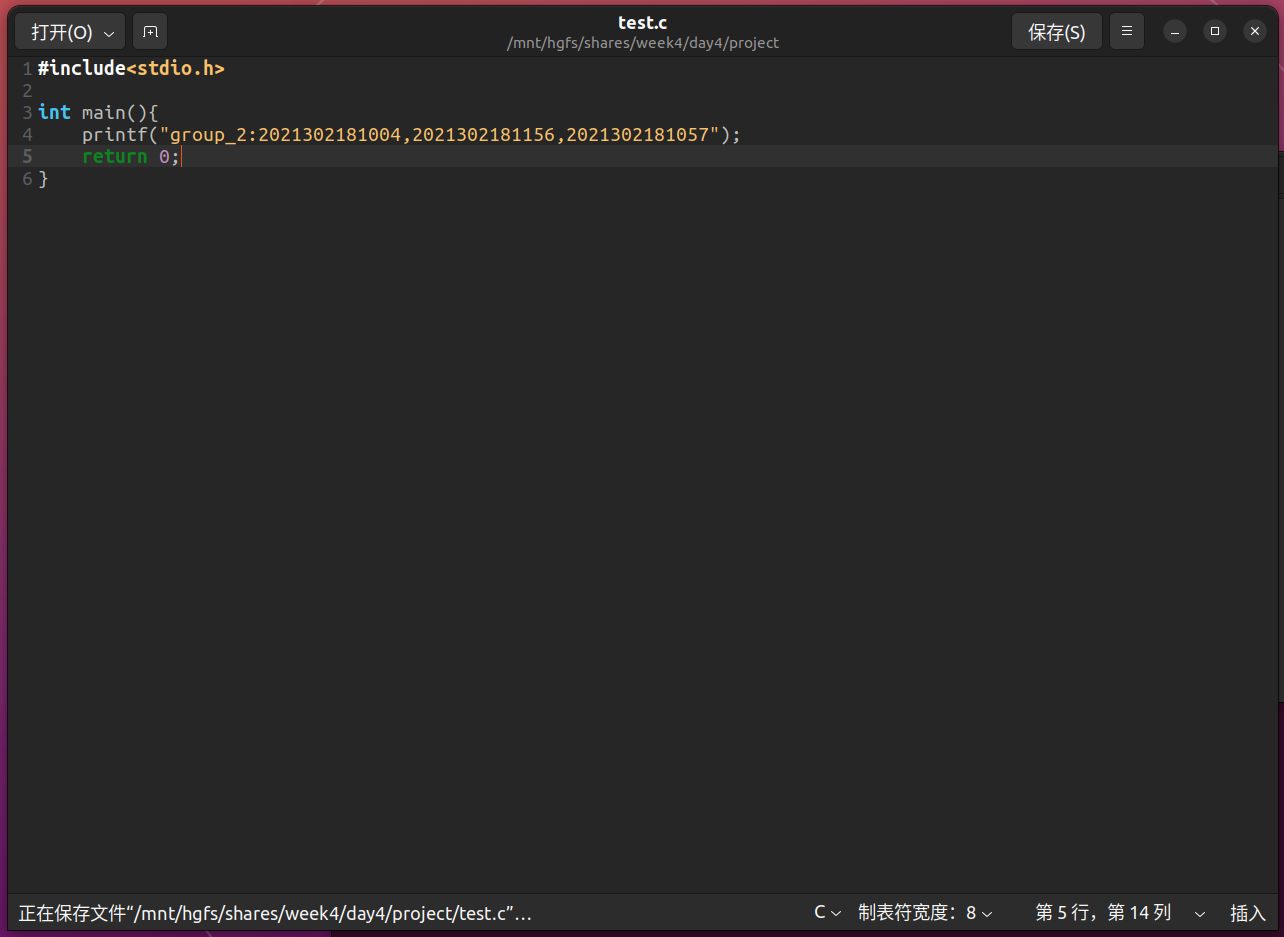
之后运行make BR\_JLEVEL=4 V=s进行编译



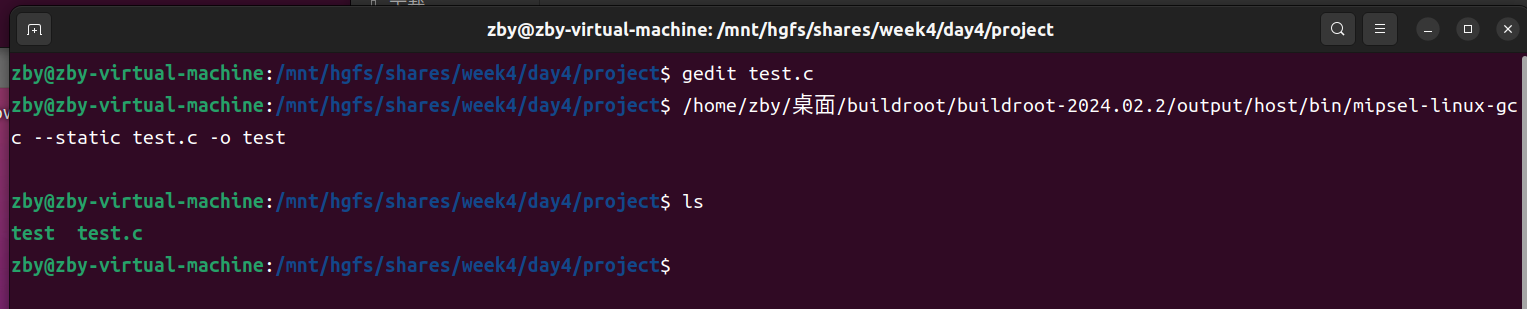
编译完成后进入到output/host/bin/，查看生成的工具如下所示



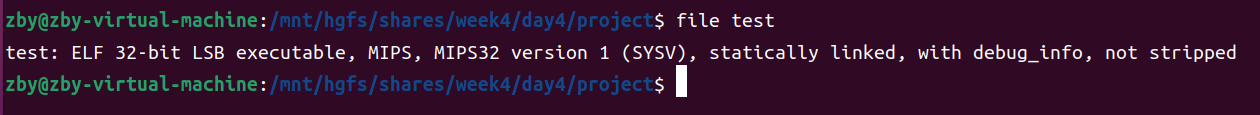
接下来新建一个项目目录，在目录中新建test.c文件如下



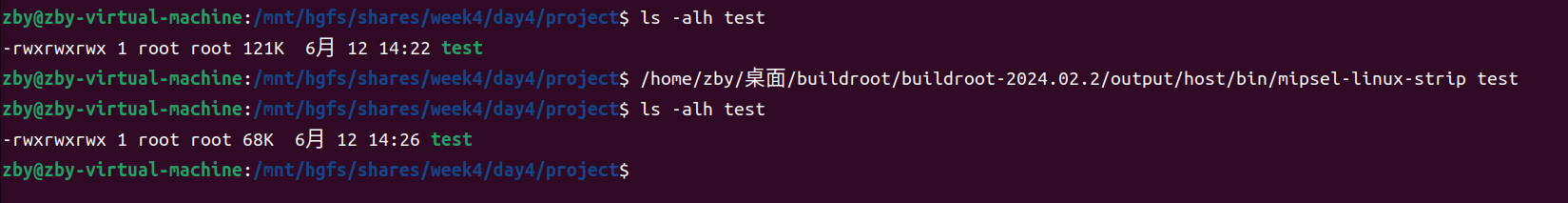
使用编译生成的gcc工具mipsel-linux-gcc对test.c文件进行编译



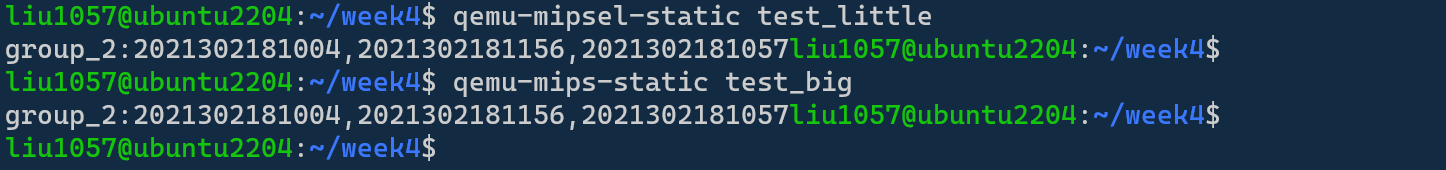
接下来查看编译生成的可执行文件的基本信息可以发现该可执行程序为mips32位小端（LSB），链接方式为静态链接



对可执行文件使用mipsel-linux-strip工具strip掉ELF可执行文件或者目标文件中不必要的调试信息，通过对比发现文件的大小减少了将近一倍

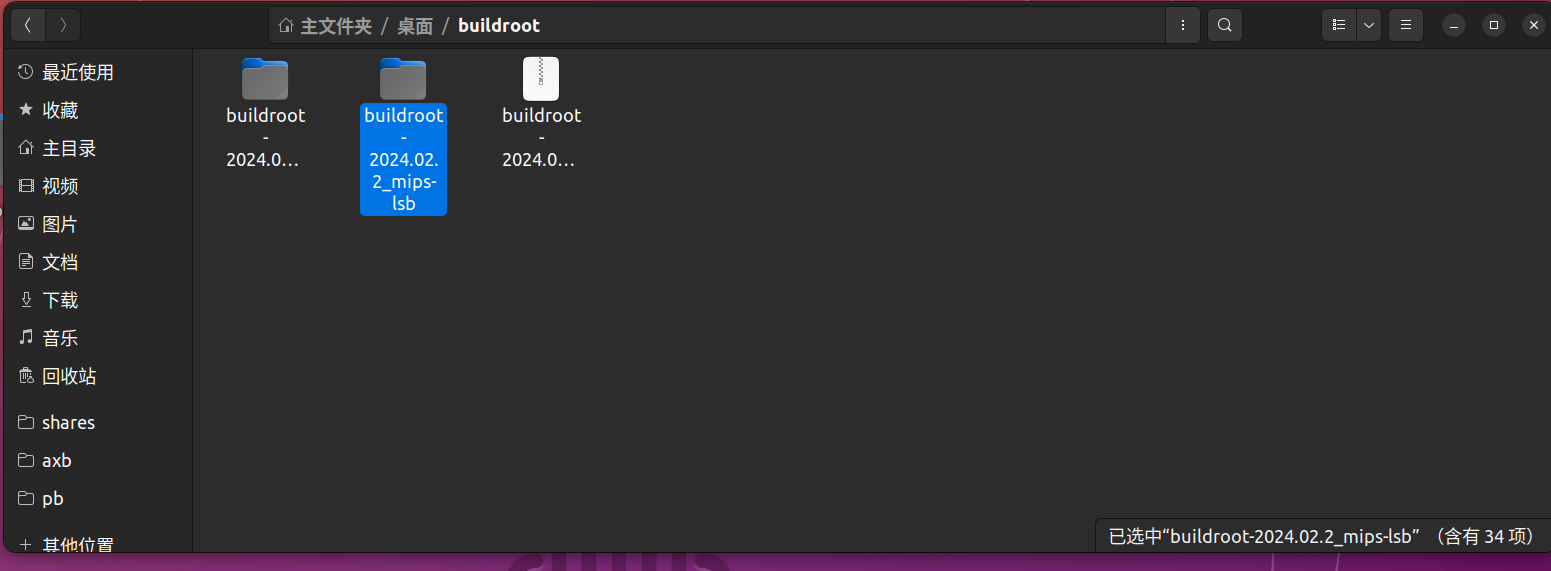


运行该可执行程序如下

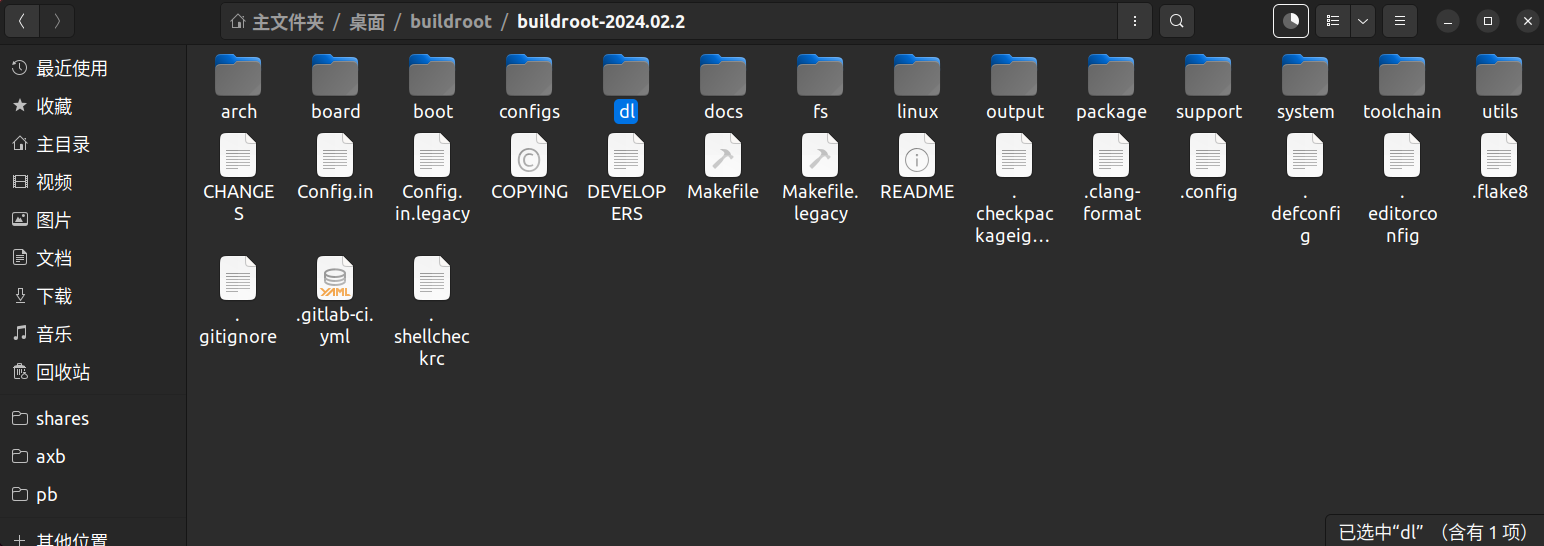


## 1.2大端

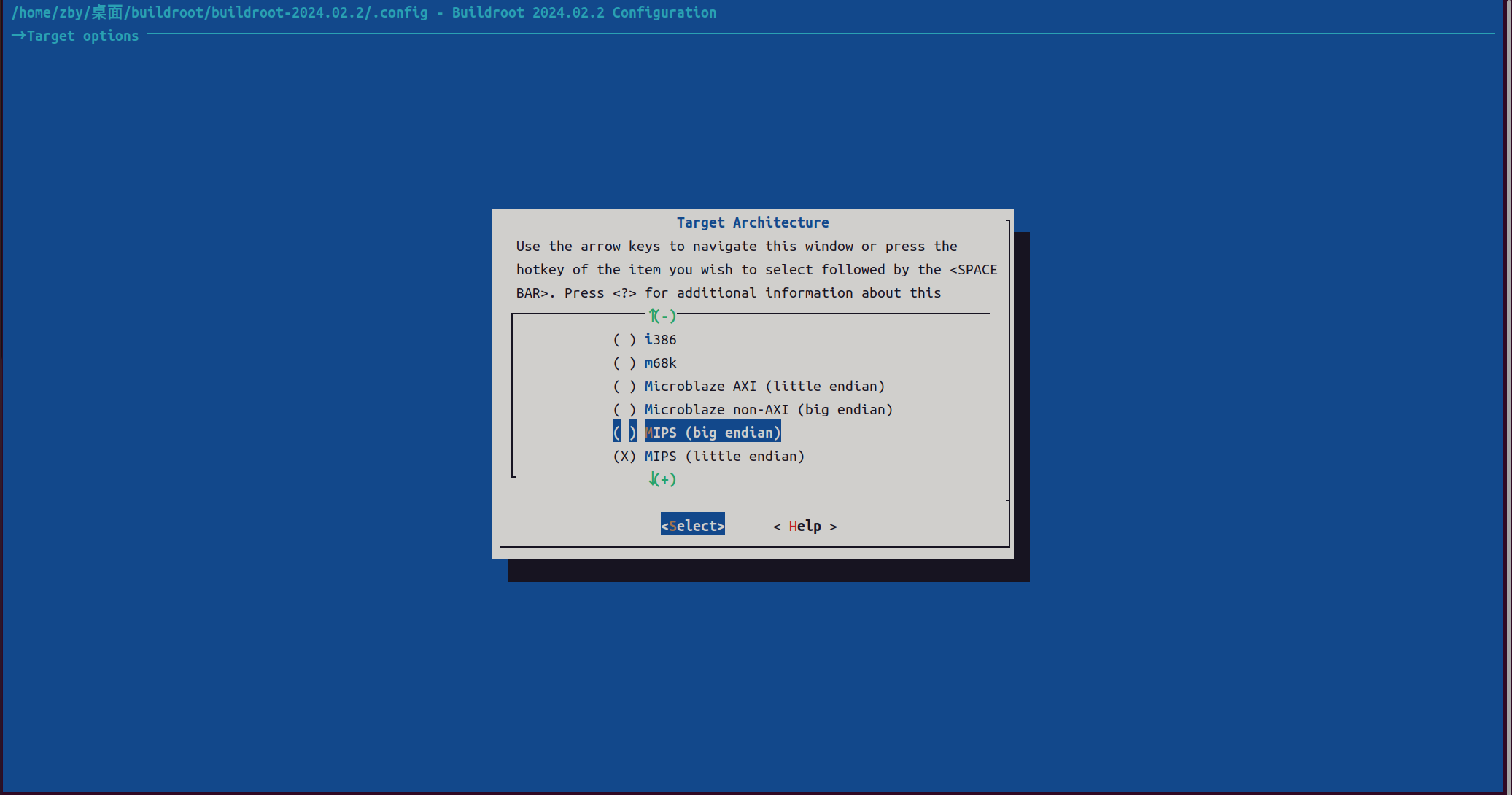
将之前的小段文件项目目录更改名称之后重新解压项目包



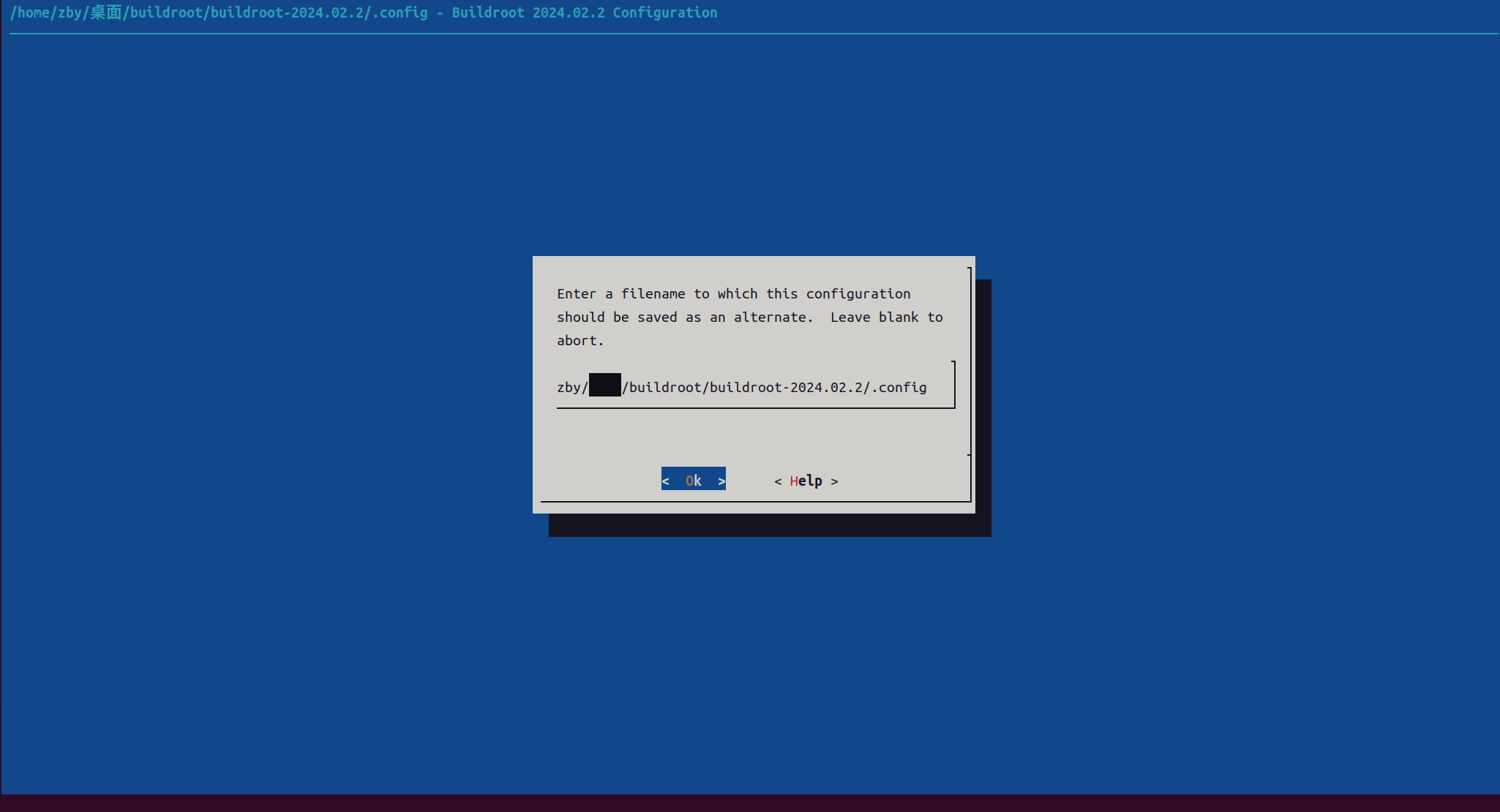
将小端操作时的dl目录拷贝到当前项目包中



之后的配置部分与小端相同，只是在选择架构时选择MIPS32大端，将目标架构更改为MIPS32大端



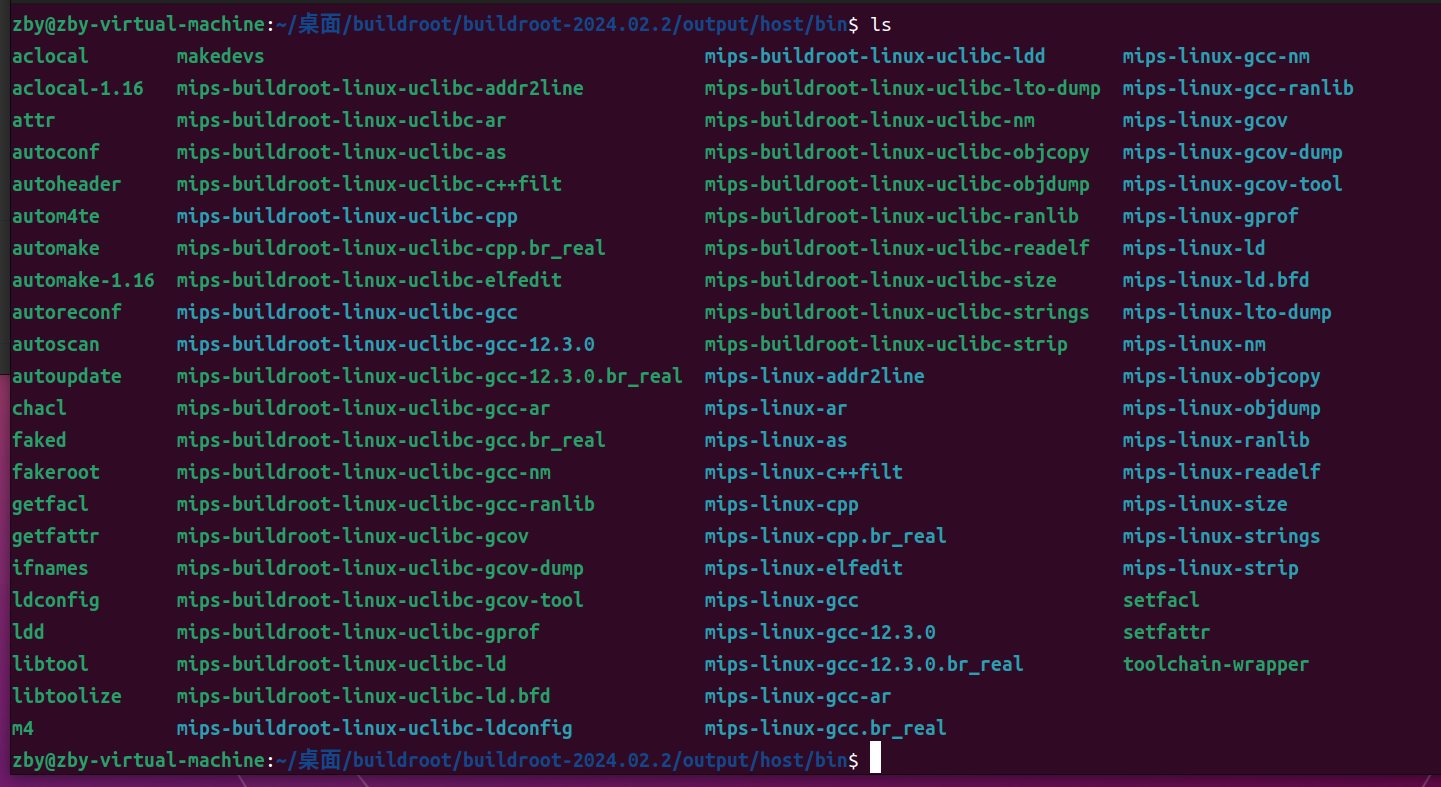
之后将配置保存



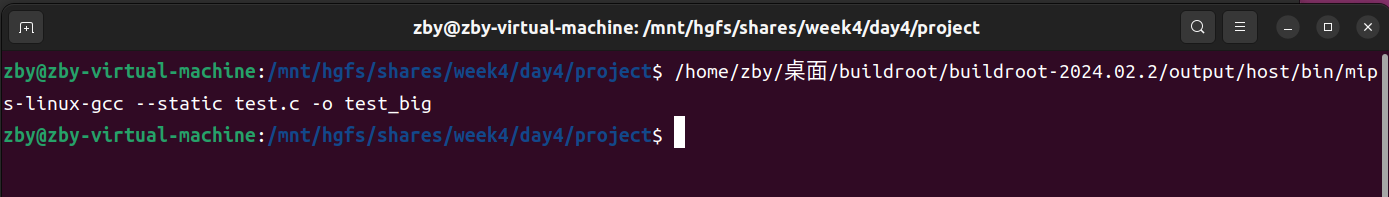
重新执行make BR\_JLEVEL=8 V=s进行编译



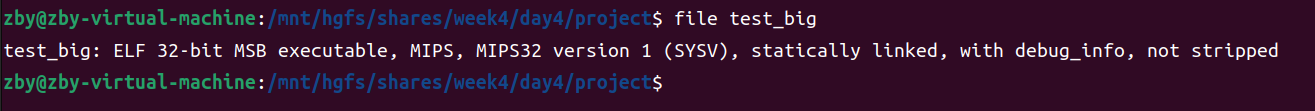
查看output/host/bin目录下的工具可以发现大端工具mips-linux-gcc



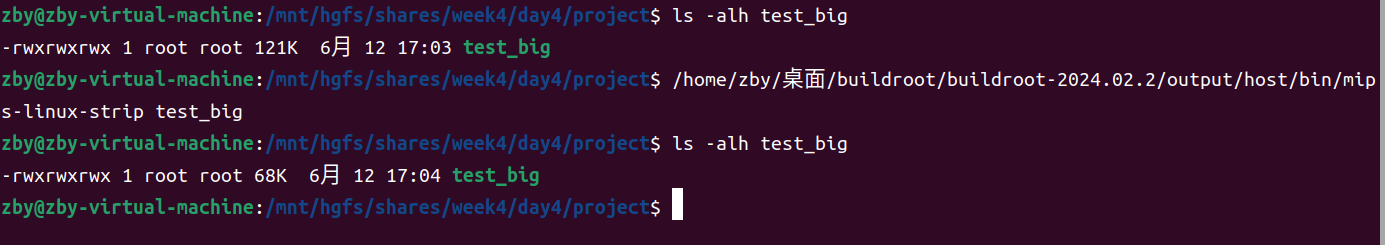
对test.c进行编译，生成可执行文件test\_big



查看该文件的基本信息如下



同样对可执行文件使用工具strip掉ELF可执行文件或者目标文件中不必要的调试信息，通过对比发现文件的大小减少了将近一倍



运行该可执行程序如下

