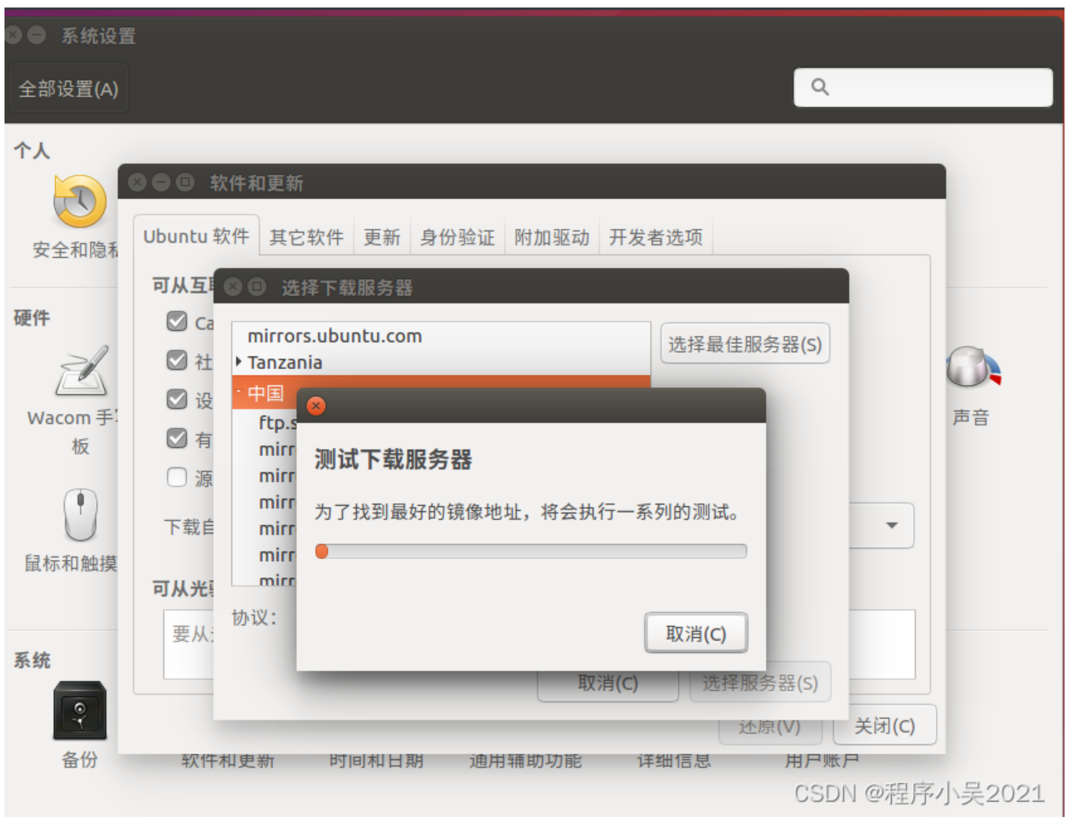
全志V3S零基础教程

文章目录

U-boot编译

环境配置

先 换源 ，避免后续的下载速度太慢



测试下载服务器

为了找到最好的镜像地址，将会执行一系列的测试。

取消(C)

执行 sudo apt-get update

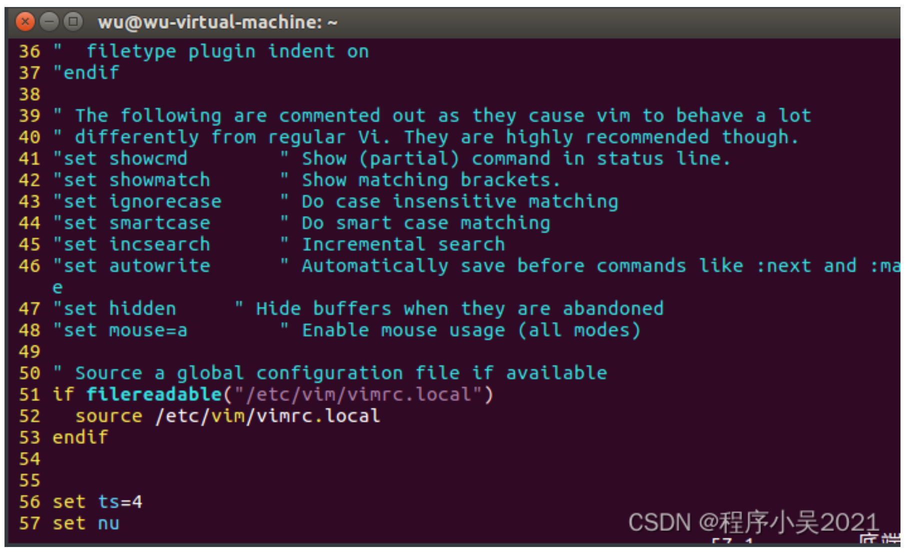
下载一个vim，等下方便修改

sudo apt-get install vim

进入vi配置文件 sudo vi /etc/vim/vimrc

在配置文件设置行距为4并且打开行号

vim的使用这里步做过多讲解，自行百度



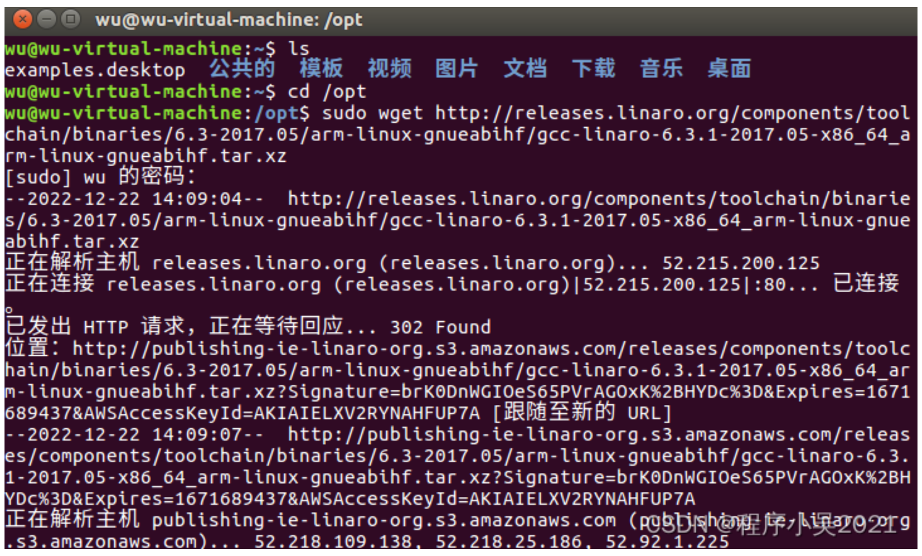
```
wu@wu-virtual-machine: ~
"  filetype plugin indent on
"endif

" The following are commented out as they cause vim to behave a lot
" differently from regular Vi. They are highly recommended though.
"set showcmd         " Show (partial) command in status line.
"set showmatch       " Show matching brackets.
"set ignorecase      " Do case insensitive matching
"set smartcase       " Do smart case matching
"set incsearch       " Incremental search
"set autowrite       " Automatically save before commands like :next and :ma
e
"set hidden      " Hide buffers when they are abandoned
"set mouse=a         " Enable mouse usage (all modes)

" Source a global configuration file if available
if filereadable("/etc/vim/vimrc.local")
  source /etc/vim/vimrc.local
endif


set ts=4
set nu
```

进入正题，配置交叉环境，安装gcc编译器 ，本教程使用的是6.3.1版本

先cd /opt文件里面

执行： sudo wget https://releases.linaro.org/components/toolchain/binaries/6.3-2017.05/arm-linux -gnueabihf/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf.tar.xz

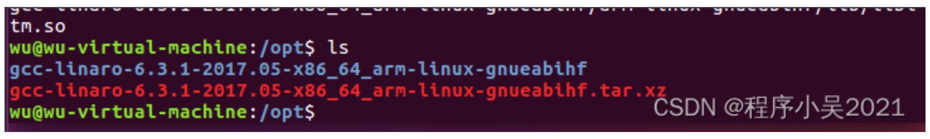
如果出现安全证书过期，请换成http进行下载



```
wu@wu-virtual-machine: /opt
wu@wu-virtual-machine:~$ ls
examples.desktop  公共的  模板  视频  图片  文档  下载  音乐  桌面
wu@wu-virtual-machine:~$ cd /opt
wu@wu-virtual-machine:/opt$ sudo wget http://releases.linaro.org/components/tool
chain/binaries/6.3-2017.05/arm-linux-gnueabihf/gcc-linaro-6.3.1-2017.05-x86_64_a
rm-linux-gnueabihf.tar.xz
[sudo] wu 的密码：
--2022-12-22 14:09:04--  http://releases.linaro.org/components/toolchain/binarie
s/6.3-2017.05/arm-linux-gnueabihf/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnue
abihf.tar.xz
正在解析主机 releases.linaro.org (releases.linaro.org)... 52.215.200.125
正在连接 releases.linaro.org (releases.linaro.org)|52.215.200.125|:80... 已连接
。
已发出 HTTP 请求，正在等待回应... 302 Found
位置：http://publishing-ie-linaro-org.s3.amazonaws.com/releases/components/toolc
hain/binaries/6.3-2017.05/arm-linux-gnueabihf/gcc-linaro-6.3.1-2017.05-x86_64_ar
m-linux-gnueabihf.tar.xz?Signature=brK0DnWGIOeS65PVrAGOxK%2BHYDc%3D&Expires=1671
689437&AWSAccessKeyId=AKIAIELXV2RYNAHFUP7A [跟随至新的 URL]
--2022-12-22 14:09:07--  http://publishing-ie-linaro-org.s3.amazonaws.com/releas
es/components/toolchain/binaries/6.3-2017.05/arm-linux-gnueabihf/gcc-linaro-6.3.
1-2017.05-x86_64_arm-linux-gnueabihf.tar.xz?Signature=brK0DnWGIOeS65PVrAGOxK%2BH
YDc%3D&Expires=1671689437&AWSAccessKeyId=AKIAIELXV2RYNAHFUP7A
正在解析主机 publishing-ie-linaro-org.s3.amazonaws.com (publishing-ie-linaro-org
.s3.amazonaws.com)... 52.218.109.138, 52.218.25.186, 52.92.1.225
```

解压交叉编译器 sudo tar xvf gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf.tar.xz

完成 ls 查看一下文件



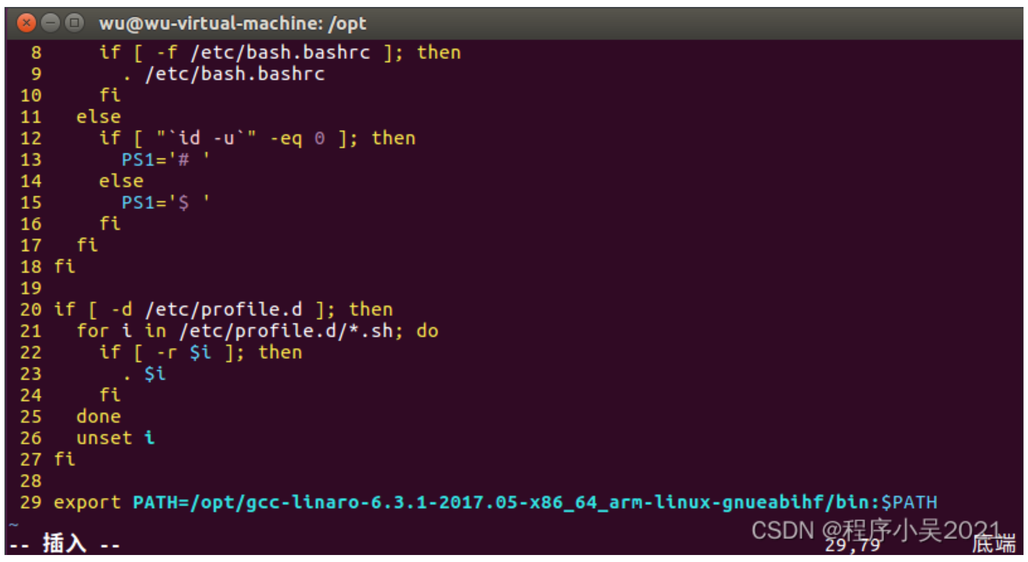
```
tm.so
wu@wu-virtual-machine:/opt$ ls
gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf
gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf.tar.xz
wu@wu-virtual-machine:/opt$
```

添加交叉编译器路径 修改profile文件

sudo vim /etc/profile

再最后一行加入export PATH=命令行路径:$PATH

如：export PATH=/opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/bin:$PATH



```
wu@wu-virtual-machine: /opt
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

export PATH=/opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/bin:$PATH
~
-- 插入 --                                                    29,79        底端
```

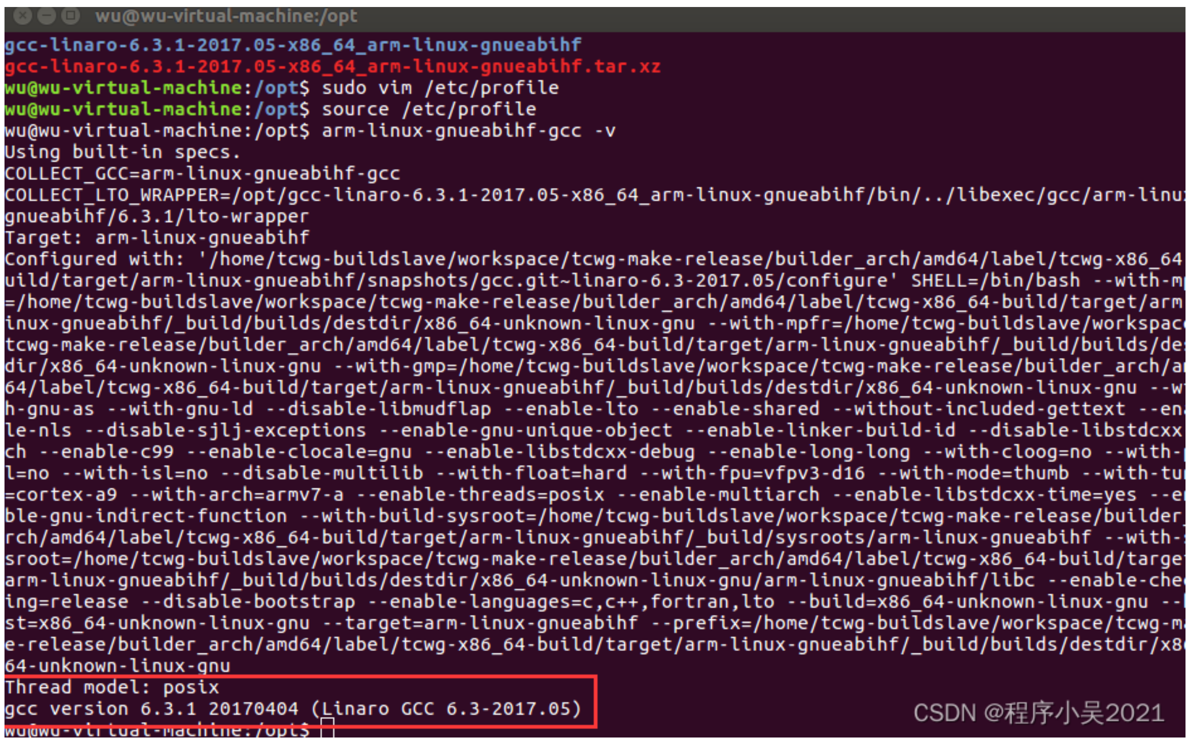
保存退出后运行source /etc/profile 生效

或者重启系统生效

备注：如果需要查看路径，终端运行 echo $PATH。 Ubuntu　几种路径添加方式对比：(11条消息) ubuntu添加环境变量的四种方法_K_K_yl的博客-CSDN博客_ubuntu添加环境变量

查看编译器是否安装成功，终端运行arm-linux-gnueabihf-gcc -v

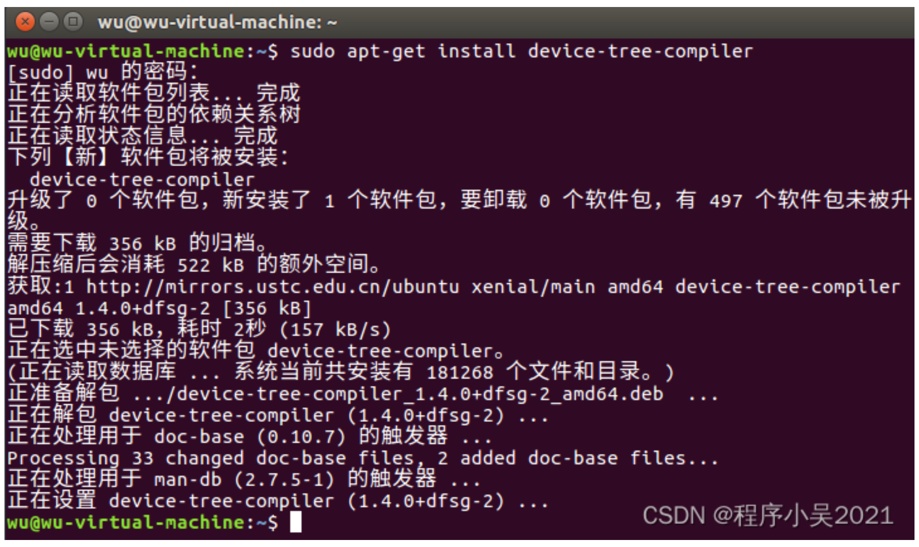
成功会返回如下：



```
wu@wu-virtual-machine:/opt
gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf
gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf.tar.xz
wu@wu-virtual-machine:/opt$ sudo vim /etc/profile
wu@wu-virtual-machine:/opt$ source /etc/profile
wu@wu-virtual-machine:/opt$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/bin/../libexec/gcc/arm-linu
gnueabihf/6.3.1/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: '/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64
uild/target/arm-linux-gnueabihf/snapshots/gcc.git~linaro-6.3-2017.05/configure' SHELL=/bin/bash --with-m
=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/arm
inux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspac
tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/arm-linux-gnueabihf/_build/builds/de
dir/x86_64-unknown-linux-gnu --with-gmp=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/a
64/label/tcwg-x86_64-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --w
h-gnu-as --with-gnu-ld --disable-libmudflap --enable-lto --enable-shared --without-included-gettext --en
le-nls --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id --disable-libstdcxx
ch --enable-c99 --enable-clocale=gnu --enable-libstdcxx-debug --enable-long-long --with-cloog=no --with-
l=no --with-isl=no --disable-multilib --with-float=hard --with-fpu=vfpv3-d16 --with-mode=thumb --with-tu
=cortex-a9 --with-arch=armv7-a --enable-threads=posix --enable-multiarch --enable-libstdcxx-time=yes --e
ble-gnu-indirect-function --with-build-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/builder
rch/amd64/label/tcwg-x86_64-build/target/arm-linux-gnueabihf/_build/sysroots/arm-linux-gnueabihf --with-
sroot=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/targe
arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu/arm-linux-gnueabihf/libc --enable-che
ing=release --disable-bootstrap --enable-languages=c,c++,fortran,lto --build=x86_64-unknown-linux-gnu --
st=x86_64-unknown-linux-gnu --target=arm-linux-gnueabihf --prefix=/home/tcwg-buildslave/workspace/tcwg-m
e-release/builder_arch/amd64/label/tcwg-x86_64-build/target/arm-linux-gnueabihf/_build/builds/destdir/x8
64-unknown-linux-gnu
Thread model: posix
gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05)
wu@wu-virtual-machine:/opt$
```

将终端退出重新打开一下 安装设备树编译器

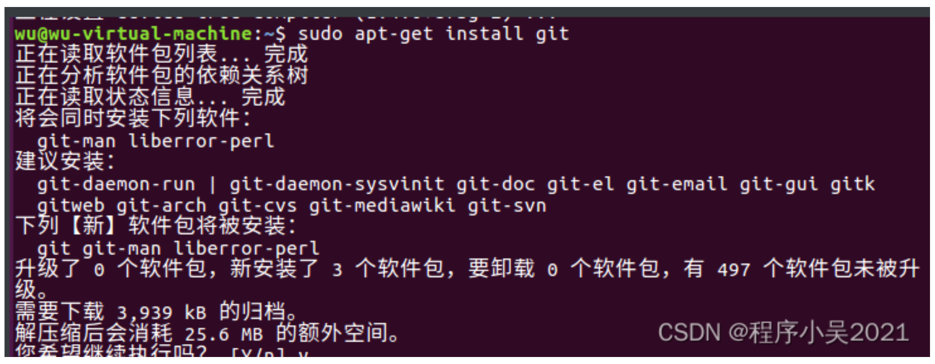
终端输入sudo apt-get install device-tree-compiler



```
wu@wu-virtual-machine: ~
wu@wu-virtual-machine:~$ sudo apt-get install device-tree-compiler
[sudo] wu 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列【新】软件包将被安装：
  device-tree-compiler
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 497 个软件包未被升
级。
需要下载 356 kB 的归档。
解压缩后会消耗 522 kB 的额外空间。
获取:1 http://mirrors.ustc.edu.cn/ubuntu xenial/main amd64 device-tree-compiler
amd64 1.4.0+dfsg-2 [356 kB]
已下载 356 kB，耗时 2秒 (157 kB/s)
正在选中未选择的软件包 device-tree-compiler。
(正在读取数据库 ... 系统当前共安装有 181268 个文件和目录。)
正准备解包 .../device-tree-compiler_1.4.0+dfsg-2_amd64.deb  ...
正在解包 device-tree-compiler (1.4.0+dfsg-2) ...
正在处理用于 doc-base (0.10.7) 的触发器 ...
Processing 33 changed doc-base files, 2 added doc-base files...
正在处理用于 man-db (2.7.5-1) 的触发器 ...
正在设置 device-tree-compiler (1.4.0+dfsg-2) ...
wu@wu-virtual-machine:~$
```

下载Uboot

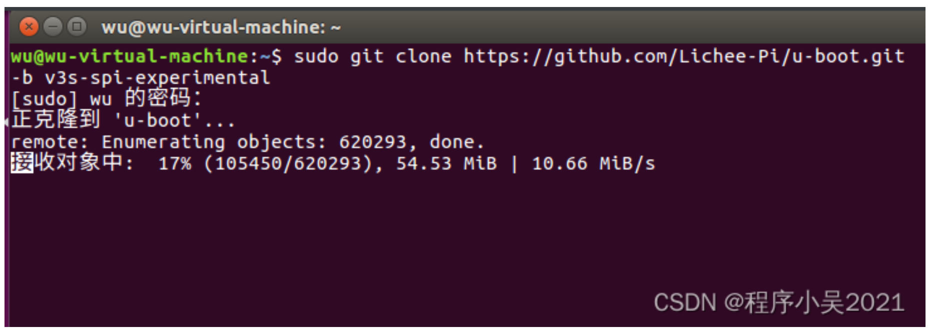
安装git，终端输入sudo apt-get install git



```
wu@wu-virtual-machine:~$ sudo apt-get install git
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件：
  git-man liberror-perl
建议安装：
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-cvs git-mediawiki git-svn
下列【新】软件包将被安装：
  git git-man liberror-perl
升级了 0 个软件包，新安装了 3 个软件包，要卸载 0 个软件包，有 497 个软件包未被升
级。
需要下载 3,939 kB 的归档。
解压缩后会消耗 25.6 MB 的额外空间。
您希望继续执行吗？ [Y/n] y
```

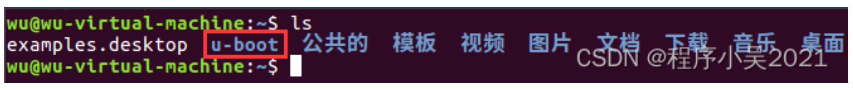
拷贝Uboot到Ubuntu下，终端输入 sudo git clone https://github.com/Lichee-Pi/u-boot.git -b v3s-spi-experimental

克隆失败，重新克隆可以解决



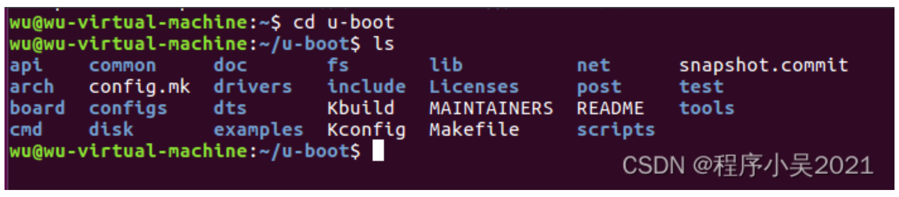
```
wu@wu-virtual-machine: ~
wu@wu-virtual-machine:~$ sudo git clone https://github.com/Lichee-Pi/u-boot.git
-b v3s-spi-experimental
[sudo] wu 的密码：
正克隆到 'u-boot'...
remote: Enumerating objects: 620293, done.
接收对象中:  17% (105450/620293), 54.53 MiB | 10.66 MiB/s
```

下载完成



```
wu@wu-virtual-machine:~$ ls
examples.desktop  u-boot  公共的  模板  视频  图片  文档  下载  音乐  桌面
wu@wu-virtual-machine:~$
```

进入u-boot



```
wu@wu-virtual-machine:~$ cd u-boot
wu@wu-virtual-machine:~/u-boot$ ls
api    common     doc       fs       lib          net      snapshot.commit
arch   config.mk  drivers   include  Licenses     post     test
board  configs    dts       Kbuild   MAINTAINERS  README   tools
cmd    disk       examples  Kconfig  Makefile     scripts
wu@wu-virtual-machine:~/u-boot$
```

uboot目录结构

├── api 存放uboot提供的API接口函数

├── arch 平台相关的部分我们只需要关心这个目录下的ARM文件夹

│ ├──arm

│ │ └──cpu

│ │ │ └──armv7

│ │ └──dts

│ │ │ └──*.dts 存放设备的dts,也就是设备配置相关的引脚信息

├── board 对于不同的平台的开发板对应的代码

├── cmd 顾名思义，大部分的命令的实现都在这个文件夹下面。

├── common 公共的代码

├── configs 各个板子的对应的配置文件都在里面，我们的Lichee配置也在里面

├── disk 对磁盘的一些操作都在这个文件夹里面，例如分区等。

├── doc 参考文档，这里面有很多跟平台等相关的使用文档。

├── drivers 各式各样的驱动文件都在这里面

├── dts 一种树形结构（device tree）这个应该是uboot新的语法

├── examples 官方给出的一些样例程序

├── fs 文件系统，uboot会用到的一些文件系统

├── include 头文件，所有的头文件都在这个文件夹下面

├── lib 一些常用的库文件在这个文件夹下面

├── Licenses 这个其实跟编译无关了，就是一些license的声明

├── net 网络相关的，需要用的小型网络协议栈

├── post 上电自检程序

├── scripts 编译脚本和Makefile文件

├── spl second program loader，即相当于二级uboot启动。

├── test 小型的单元测试程序。

└── tools 里面有很多uboot常用的工具。

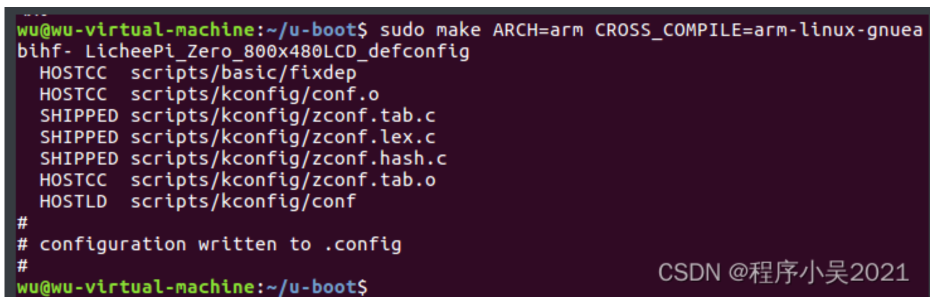
安装make 终端输入 sudo apt-get install make

设置参数

sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- LicheePi_Zero_800x480LCD_defconfig

#or sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- LicheePi_Zero480x272LCD_defconfig

#or sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- LicheePi_Zero_defconfig



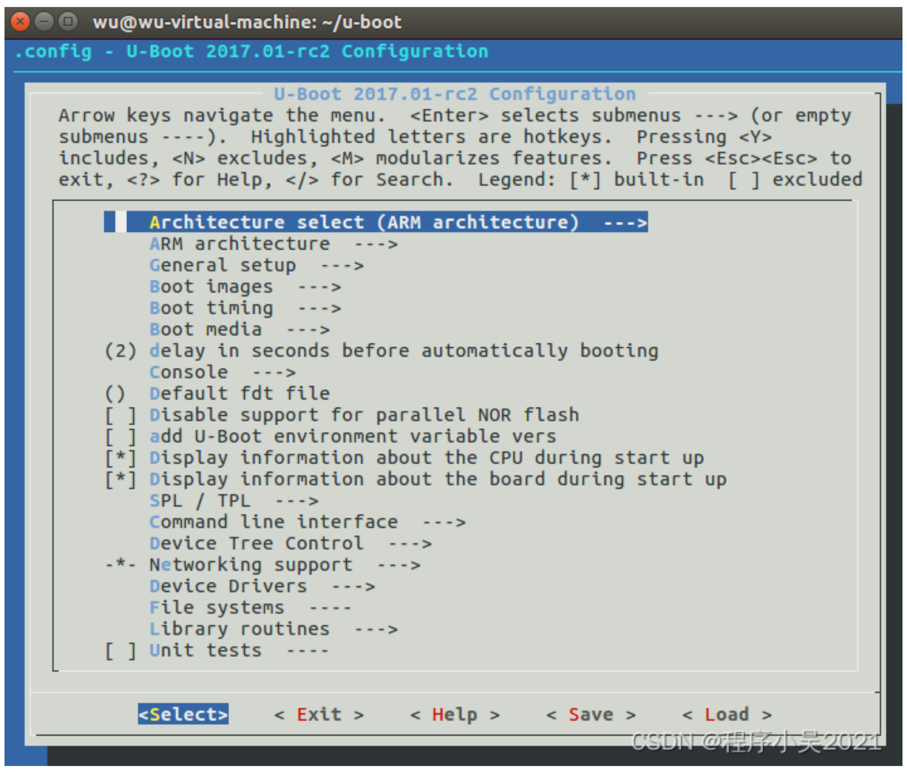
```
wu@wu-virtual-machine:~/u-boot$ sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnuea
bihf- LicheePi_Zero_800x480LCD_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
wu@wu-virtual-machine:~/u-boot$
```

安装libncurses5-dev，终端输入sudo apt-get install libncurses5-dev解决

uboot配置命令

终端输入 sudo make menuconfig

此时会跳出以下配置界面



```
wu@wu-virtual-machine: ~/u-boot
.config - U-Boot 2017.01-rc2 Configuration
                 U-Boot 2017.01-rc2 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

        Architecture select (ARM architecture)  --->
        ARM architecture  --->
        General setup  --->
        Boot images  --->
        Boot timing  --->
        Boot media  --->
    (2) delay in seconds before automatically booting
        Console  --->
    ()  Default fdt file
    [ ] Disable support for parallel NOR flash
    [ ] add U-Boot environment variable vers
    [*] Display information about the CPU during start up
    [*] Display information about the board during start up
        SPL / TPL  --->
        Command line interface  --->
        Device Tree Control  --->
    -*- Networking support  --->
        Device Drivers  --->
        File systems  ----
        Library routines  --->
    [ ] Unit tests  ----

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

—按回车，即选择当前菜单

------- 按Y 代表该config选项选中

------- 按N 代表不选中该选项

-------- 按M

代表该驱动编译成**.ko的方式，在系统起来之后，当驱动需要的时候加载---------按/ 可以查找某个选项---------退出<** > ----------按Y选中后的状态

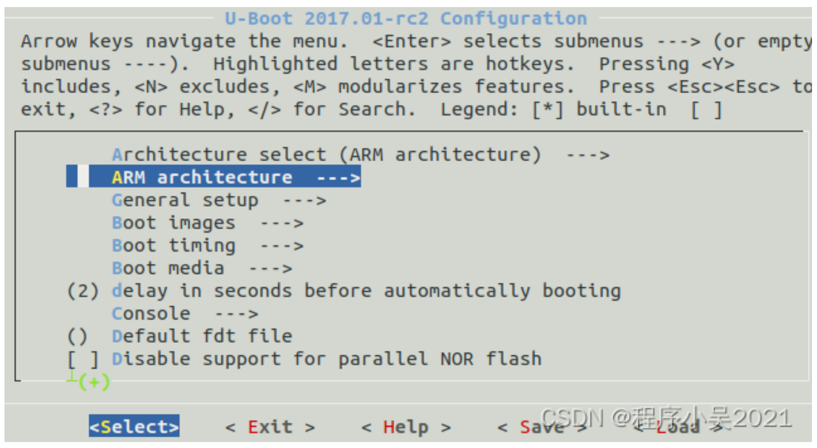
这里面有几个常见的配置选项我们可以看下:

1. 第一个Architecture select架构选择，不用质疑这个是ARM架构
2. 第二个ARM architecture

这个选项比较重要，主要配置ARM框架下的常用的配置函数以及LCD等参数

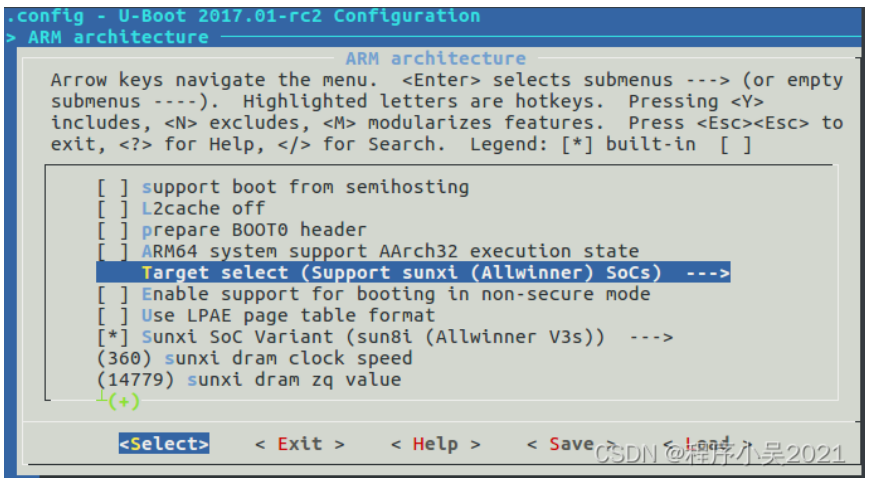
DDR配置

选择这个，进入



```
                 U-Boot 2017.01-rc2 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  [ ]

        Architecture select (ARM architecture)  --->
        ARM architecture  --->
        General setup  --->
        Boot images  --->
        Boot timing  --->
        Boot media  --->
    (2) delay in seconds before automatically booting
        Console  --->
    ()  Default fdt file
    [ ] Disable support for parallel NOR flash
    ↓(+)

       <Select>    < Exit >    < Help >    < Save >    < Load >
```

再进这个



```
.config - U-Boot 2017.01-rc2 Configuration
> ARM architecture
                              ARM architecture
  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  [ ]

      [ ] support boot from semihosting
      [ ] L2cache off
      [ ] prepare BOOT0 header
      [ ] ARM64 system support AArch32 execution state
          Target select (Support sunxi (Allwinner) SoCs)  --->
      [ ] Enable support for booting in non-secure mode
      [ ] Use LPAE page table format
      [*] Sunxi SoC Variant (sun8i (Allwinner V3s))  --->
      (360) sunxi dram clock speed
      (14779) sunxi dram zq value
      ↓(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

```
···
Target select (Support sunxi (Allwinner) SoCs)   #进去之后可以选择sunxi Soc系列芯片
···
[*] Sunxi SoC Variant    # 这个就是对芯片Soc 的选择，我们可以看到配置选择了`sun8i (Allwinner V3s)
(360) sunxi dram clock speed     # 配置dram的时钟速率
(14779) sunxi dram zq value     #  配置dram的ZQ值，是用来动态加强DDR3的
-*- Board uses DDR2 DRAM     #   使用DDR2 DRAM
```

时钟频率配置#

`Boot images --->(1008000000) CPU clock frequency`

这里设置了CPU的时钟频率

开机延时设置#

`delay in seconds before automatically booting`

这个是uboot开机的时候的一个等待时间的秒数，可以改大一点，默认是2s

SPL配置

```
SPL / TPL ---> 这个就是SPL相关的配置了
[*]   MMC raw mode: by sector                         按扇区
(0x50)  Address on the MMC to load U-Boot from  mmc加载uboot的地址
[*] Support GPIO                                   支持GPIO
[ ] Support I2C                                   支持I2C
[*] Support common libraries                      支持通用lib
[*] Support disk paritions                        支持分区
[*] Support generic libraries                     支持一般lib库
[*] Support MMC                                   支持MMC
[*] Support power drivers                    支持电源驱动
[*] Support serial                                 支持串口
```

spi lcd及串口输出配置

修改include/configs/sun8i.h 使uboot从tf卡启动

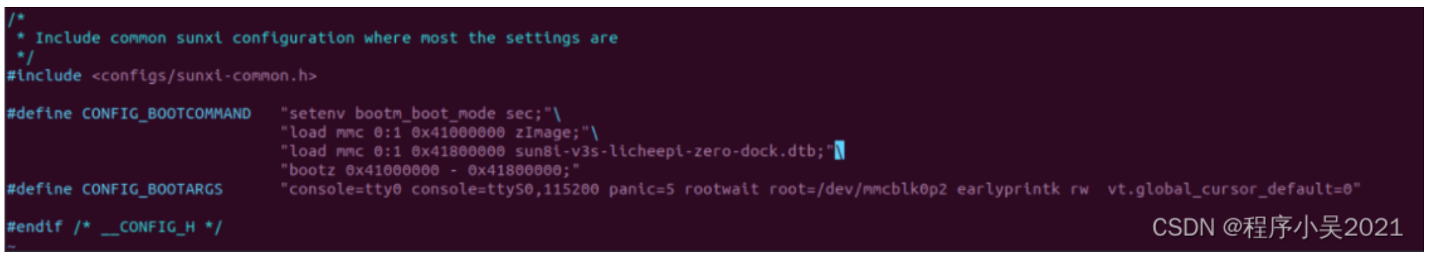
不需要LCD的在sun8i.h中加入以下

```
#define CONFIG_BOOTCOMMAND   "setenv bootm_boot_mode sec; "
                           "load mmc 0:1 0x41000000 zImage; "
                            "load mmc 0:1 0x41800000 sun8i-v3s-licheepi-zero-dock.dtb; "
                            "bootz 0x41000000 - 0x41800000;"
\#define CONFIG_BOOTARGS     "console=ttyS0,115200 panic=5 rootwait root=/dev/mmcblk0p2 earlyprintk rw  vt.global_cursor_default=0"
```

需要LCD显示的在sun8i.h中加入以下

```
#define CONFIG_BOOTCOMMAND   "setenv bootm_boot_mode sec; " \
                           "load mmc 0:1 0x41000000 zImage; " \
                           "load mmc 0:1 0x41800000 sun8i-v3s-licheepi-zero-dock.dtb; " \
                           "bootz 0x41000000 - 0x41800000;"
#define CONFIG_BOOTARGS     "console=tty0 console=ttyS0,115200 panic=5 rootwait root=/dev/mmcblk0p2 earlyprintk rw  vt.global_cursor_default=0"
```

备注：因为我们需要使用lcd显示跟串口同时输出，所以使用第二种加入后如下



```
/*
 * Include common sunxi configuration where most the settings are
 */
#include <configs/sunxi-common.h>

#define CONFIG_BOOTCOMMAND   "setenv bootm_boot_mode sec;"\
                             "load mmc 0:1 0x41000000 zImage;"\
                             "load mmc 0:1 0x41800000 sun8i-v3s-licheepi-zero-dock.dtb;"\
                             "bootz 0x41000000 - 0x41800000;"
#define CONFIG_BOOTARGS      "console=tty0 console=ttyS0,115200 panic=5 rootwait root=/dev/mmcblk0p2 earlyprintk rw  vt.global_cursor_default=0"

#endif /* __CONFIG_H */
```

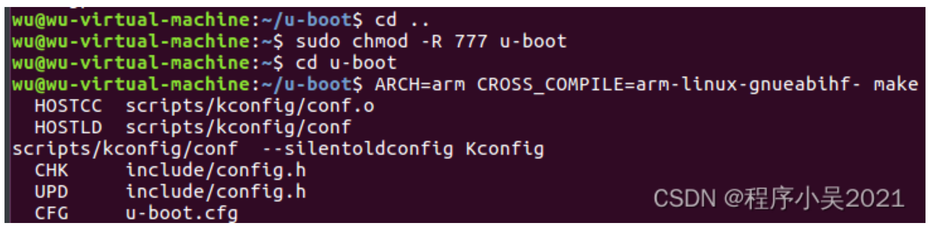
执行编译

`ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make`

编译成功后可以uboot目录发现多出了个u-boot-sunxi-with-spl.bin

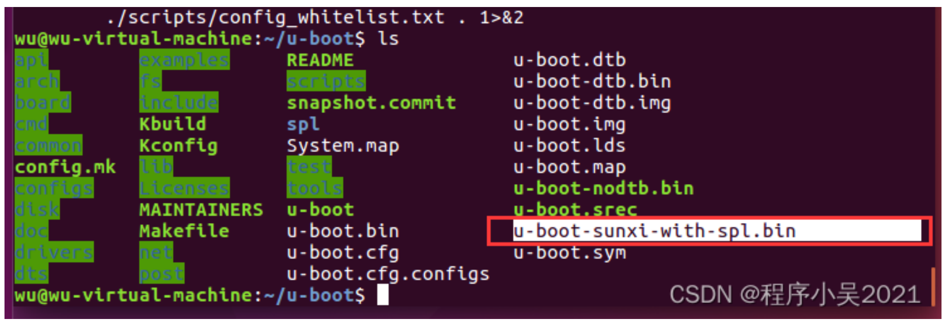
如果权限不够，可以给这个文件夹最高权限

sudo chmod -R 777 u-boot



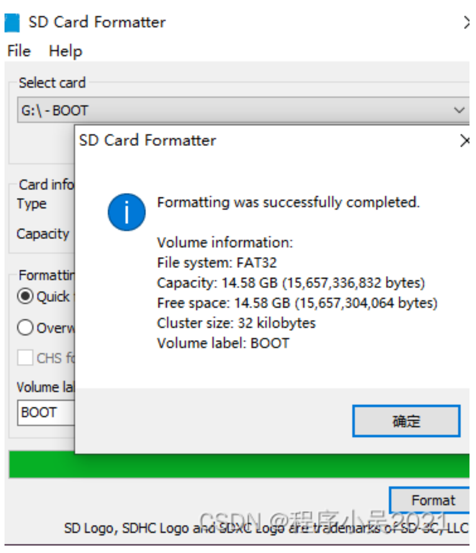
```
wu@wu-virtual-machine:~/u-boot$ cd ..
wu@wu-virtual-machine:~$ sudo chmod -R 777 u-boot
wu@wu-virtual-machine:~$ cd u-boot
wu@wu-virtual-machine:~/u-boot$ ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make
  HOSTCC  scripts/kconfig/conf.o
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf  --silentoldconfig Kconfig
  CHK     include/config.h
  UPD     include/config.h
  CFG     u-boot.cfg
```

编译成功后可以uboot目录发现多出了个u-boot-sunxi-with-spl.bin



```
        ./scripts/config_whitelist.txt . 1>&2
wu@wu-virtual-machine:~/u-boot$ ls
api          examples     README              u-boot.dtb
arch         fs           scripts             u-boot-dtb.bin
board        include      snapshot.commit     u-boot-dtb.img
cmd          Kbuild       spl                 u-boot.img
common       Kconfig      System.map          u-boot.lds
config.mk    lib          test                u-boot.map
configs      Licenses     tools               u-boot-nodtb.bin
disk         MAINTAINERS  u-boot              u-boot.srec
doc          Makefile     u-boot.bin          u-boot-sunxi-with-spl.bin
drivers      net          u-boot.cfg          u-boot.sym
dts          post         u-boot.cfg.configs
wu@wu-virtual-machine:~/u-boot$
```

然后需要将SD卡格式化



SD Card Formatter

File Help

Select card: G:\ - BOOT

Formatting was successfully completed.

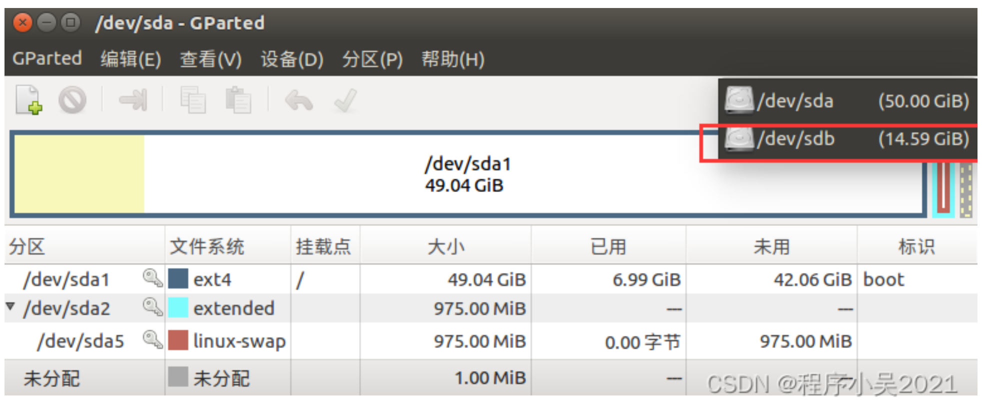
Volume information:
File system: FAT32
Capacity: 14.58 GB (15,657,336,832 bytes)
Free space: 14.58 GB (15,657,304,064 bytes)
Cluster size: 32 kilobytes
Volume label: BOOT

Card info
Type
Capacity
Quick
Overw
CHS f
Volume la
BOOT
确定
Format
SD Logo, SDHC Logo and SDXC Logo are trademarks of SD-3C, LLC

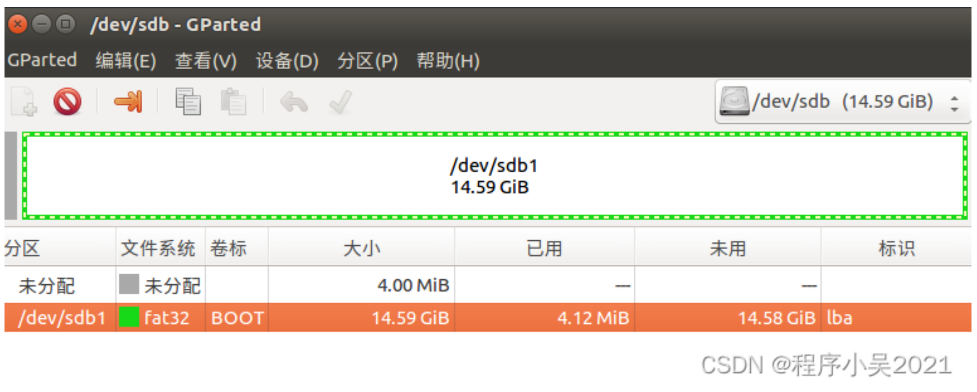
分区需要在Ubuntu进行，先安装软件

sudo apt-get install gparted

打开查看SD卡是sdb

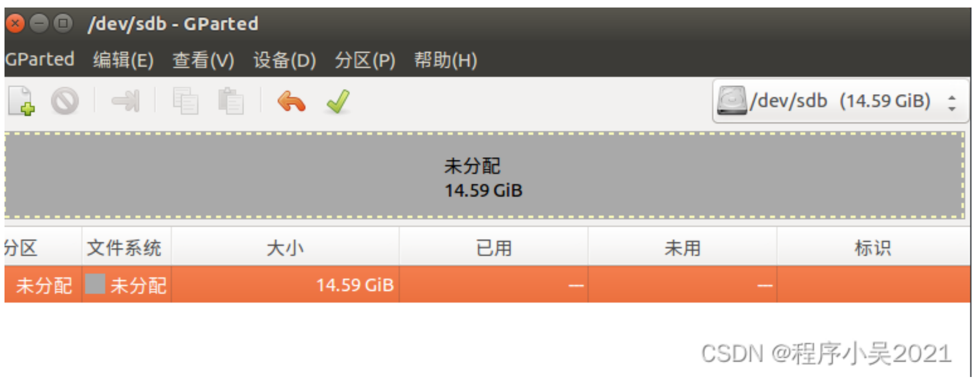


分区	文件系统	挂载点	大小	已用	未用	标识
/dev/sda1	ext4	/	49.04 GiB	6.99 GiB	42.06 GiB	boot
▾ /dev/sda2	extended		975.00 MiB	—	—	
/dev/sda5	linux-swap		975.00 MiB	0.00 字节	975.00 MiB	
未分配	未分配		1.00 MiB	—		



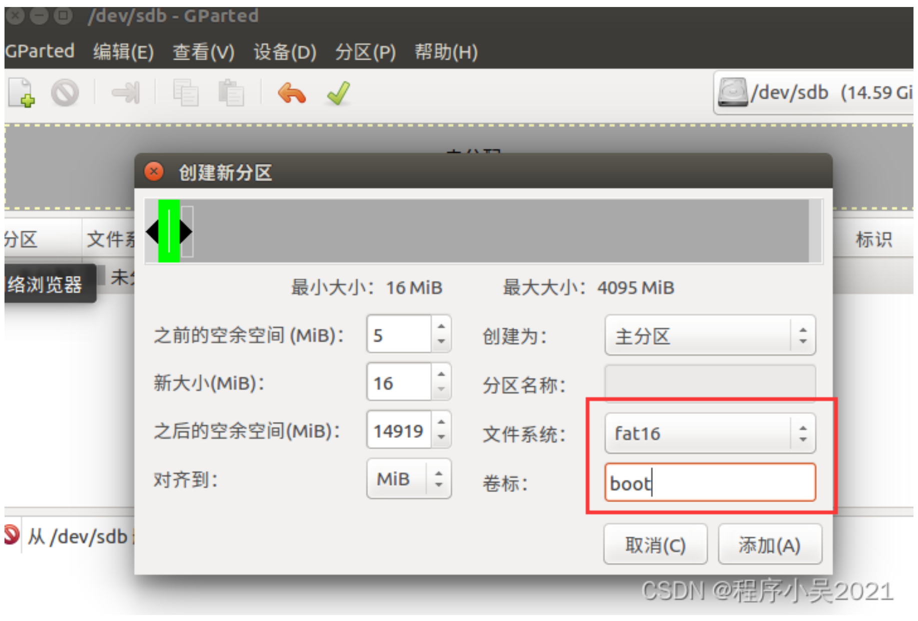
分区	文件系统	卷标	大小	已用	未用	标识
未分配	未分配		4.00 MiB	—	—	
/dev/sdb1	fat32	BOOT	14.59 GiB	4.12 MiB	14.58 GiB	lba

右键删除fat32



分区	文件系统	大小	已用	未用	标识
未分配	未分配	14.59 GiB	—	—	

分区一（boot分区）设置：


/dev/sdb - GParted

GParted 编辑(E) 查看(V) 设备(D) 分区(P) 帮助(H)

/dev/sdb (14.59 Gi

创建新分区

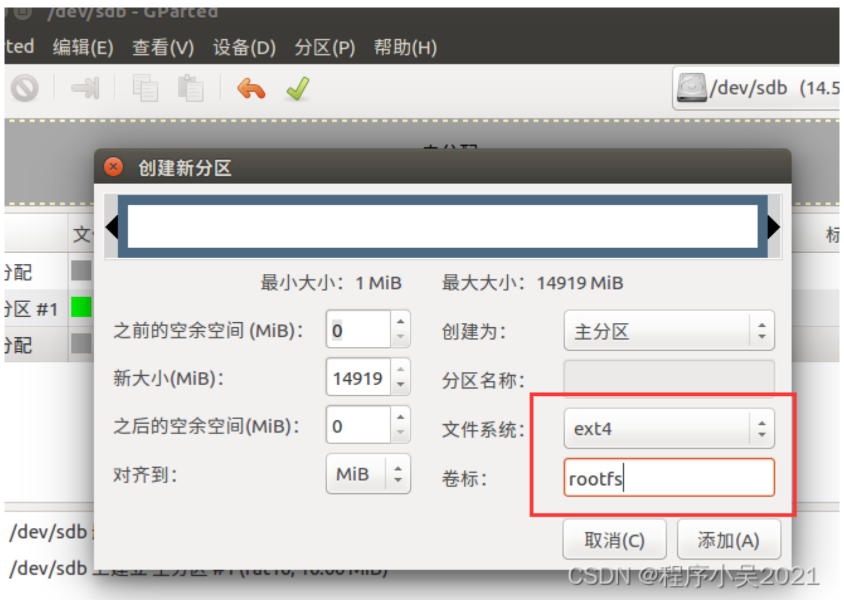
最小大小: 16 MiB 最大大小: 4095 MiB

之前的空余空间 (MiB): 5
新大小(MiB): 16
之后的空余空间(MiB): 14919
对齐到: MiB

创建为: 主分区
分区名称:
文件系统: fat16
卷标: boot

取消(C) 添加(A)

分区二（rootfs分区）设置：


/dev/sdb - GParted

编辑(E) 查看(V) 设备(D) 分区(P) 帮助(H)

创建新分区

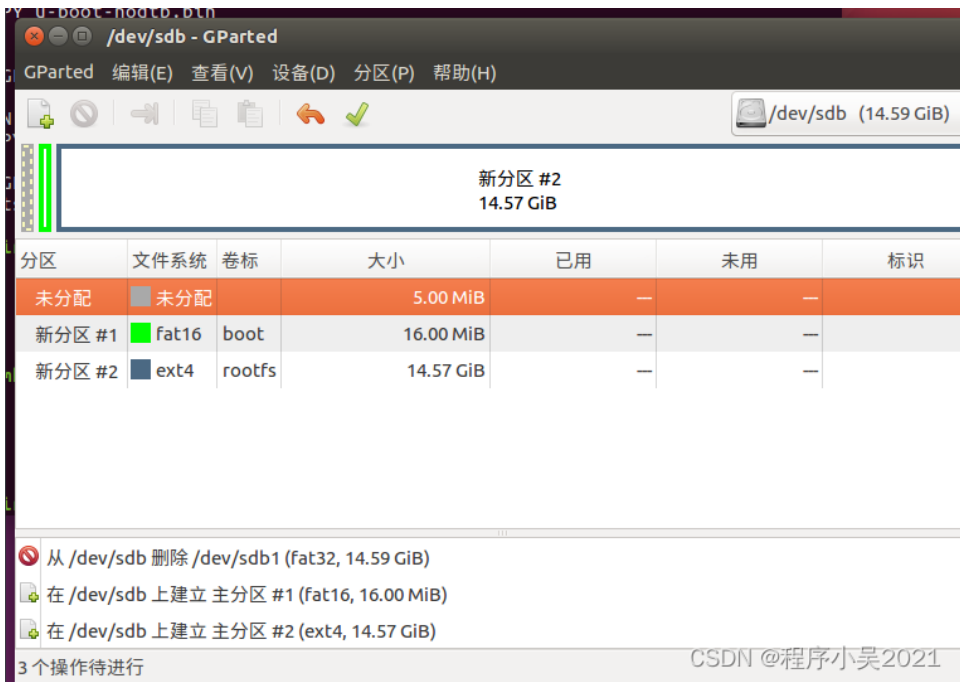
最小大小: 1 MiB 最大大小: 14919 MiB

之前的空余空间 (MiB): 0
新大小(MiB): 14919
之后的空余空间(MiB): 0
对齐到: MiB

创建为: 主分区
分区名称:
文件系统: ext4
卷标: rootfs

取消(C) 添加(A)

最终得到


/dev/sdb - GParted

GParted 编辑(E) 查看(V) 设备(D) 分区(P) 帮助(H)

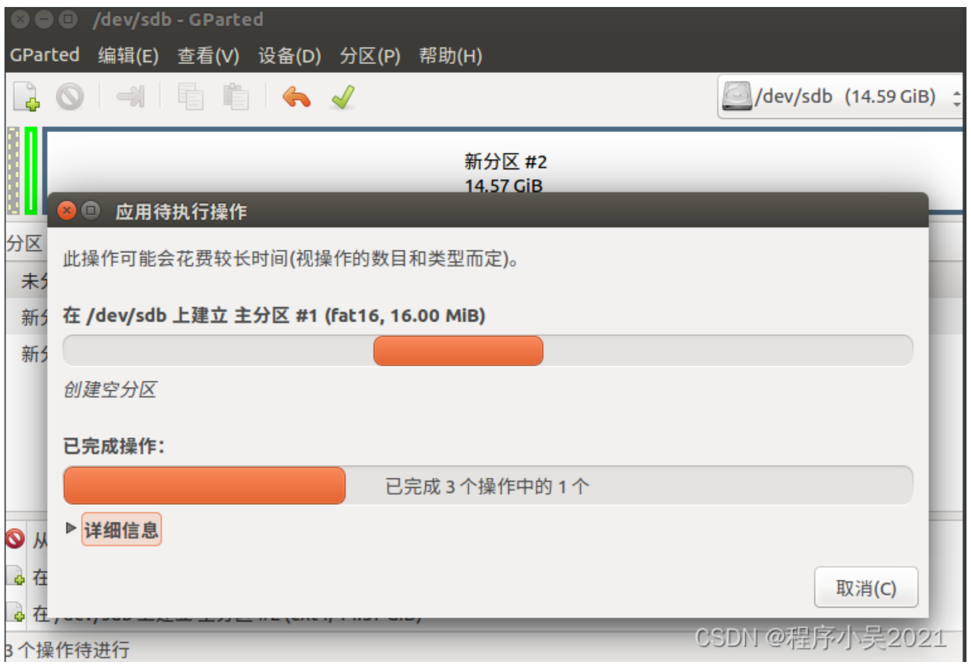
/dev/sdb (14.59 GiB)

新分区 #2
14.57 GiB

分区	文件系统	卷标	大小	已用	未用	标识
未分配	未分配		5.00 MiB	—	—	
新分区 #1	🟩 fat16	boot	16.00 MiB	—	—	
新分区 #2	■ ext4	rootfs	14.57 GiB	—	—	

从 /dev/sdb 删除 /dev/sdb1 (fat32, 14.59 GiB)
在 /dev/sdb 上建立 主分区 #1 (fat16, 16.00 MiB)
在 /dev/sdb 上建立 主分区 #2 (ext4, 14.57 GiB)
3 个操作待进行

点一下勾勾完成，耐心等待


/dev/sdb - GParted

GParted 编辑(E) 查看(V) 设备(D) 分区(P) 帮助(H)

/dev/sdb (14.59 GiB)

应用待执行操作

此操作可能会花费较长时间(视操作的数目和类型而定)。

在 /dev/sdb 上建立 主分区 #1 (fat16, 16.00 MiB)

创建空分区

已完成操作:

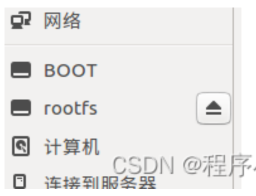
已完成 3 个操作中的 1 个

详细信息

取消(C)

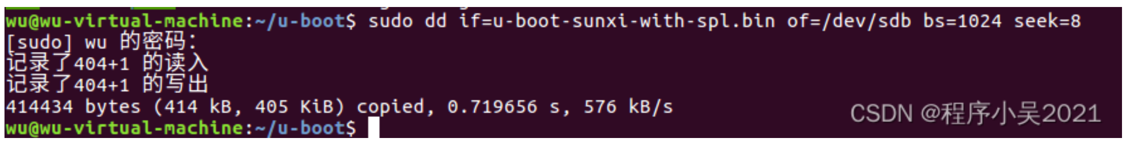
3 个操作待进行

完成后拔插一下这样就分区成功



烧录到SD卡的8k偏移处

sudo dd if=u-boot-sunxi-with-spl.bin of=/dev/sdb bs=1024 seek=8



```
wu@wu-virtual-machine:~/u-boot$ sudo dd if=u-boot-sunxi-with-spl.bin of=/dev/sdb bs=1024 seek=8
[sudo] wu 的密码:
记录了404+1 的读入
记录了404+1 的写出
414434 bytes (414 kB, 405 KiB) copied, 0.719656 s, 576 kB/s
wu@wu-virtual-machine:~/u-boot$
```

Kernel 编译

1、安装依赖#

sudo apt install git wget make gcc flex bison libssl-dev bc kmod

2、下载Linux源码

git clone -b zero-5.2.y --depth 1 https://github.com/Lichee-Pi/linux.git

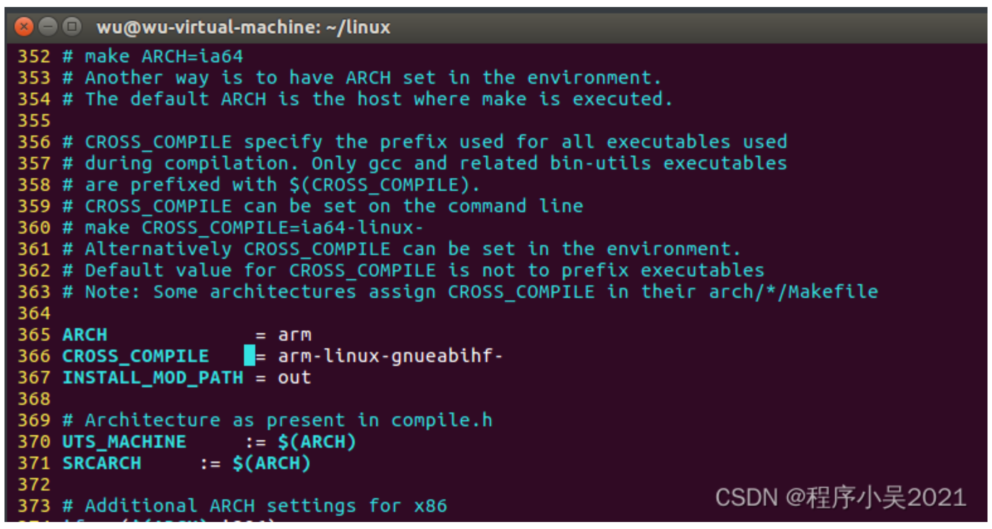
3、编译源码

修改Makefile文件

cd linux

vim Makefile

```
[364]行 修改为ARCH             = arm
[365]行 添加 CROSS_COMPILE    = arm-linux-gnueabihf-
[366]行 添加 INSTALL_MOD_PATH = out
```

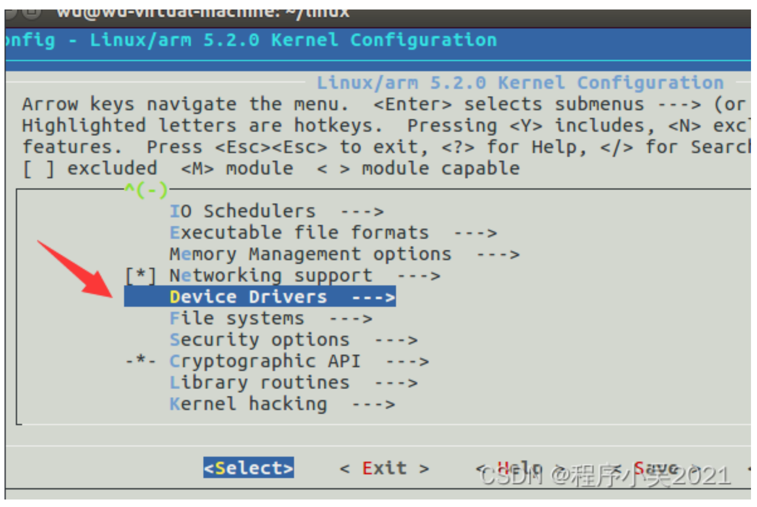


```
wu@wu-virtual-machine: ~/linux
352 # make ARCH=ia64
353 # Another way is to have ARCH set in the environment.
354 # The default ARCH is the host where make is executed.
355
356 # CROSS_COMPILE specify the prefix used for all executables used
357 # during compilation. Only gcc and related bin-utils executables
358 # are prefixed with $(CROSS_COMPILE).
359 # CROSS_COMPILE can be set on the command line
360 # make CROSS_COMPILE=ia64-linux-
361 # Alternatively CROSS_COMPILE can be set in the environment.
362 # Default value for CROSS_COMPILE is not to prefix executables
363 # Note: Some architectures assign CROSS_COMPILE in their arch/*/Makefile
364
365 ARCH              = arm
366 CROSS_COMPILE     = arm-linux-gnueabihf-
367 INSTALL_MOD_PATH = out
368
369 # Architecture as present in compile.h
370 UTS_MACHINE      := $(ARCH)
371 SRCARCH      := $(ARCH)
372
373 # Additional ARCH settings for x86
```

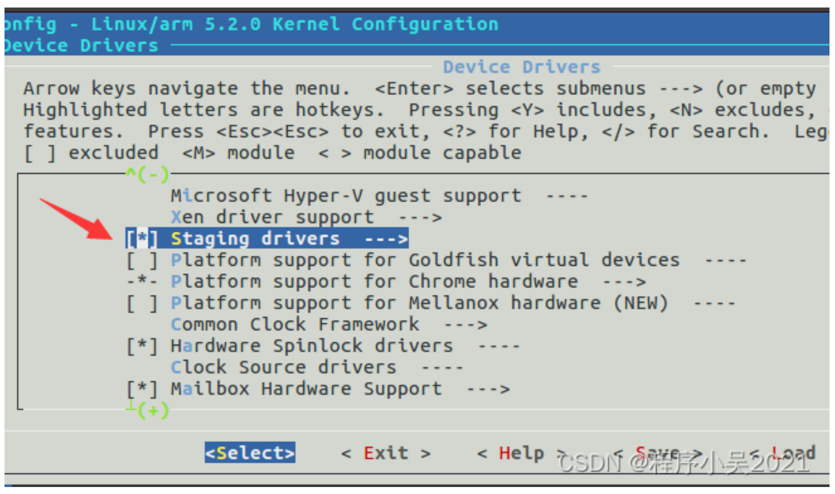
修改参数：添加LCD st7789

终端输入make menuconfig

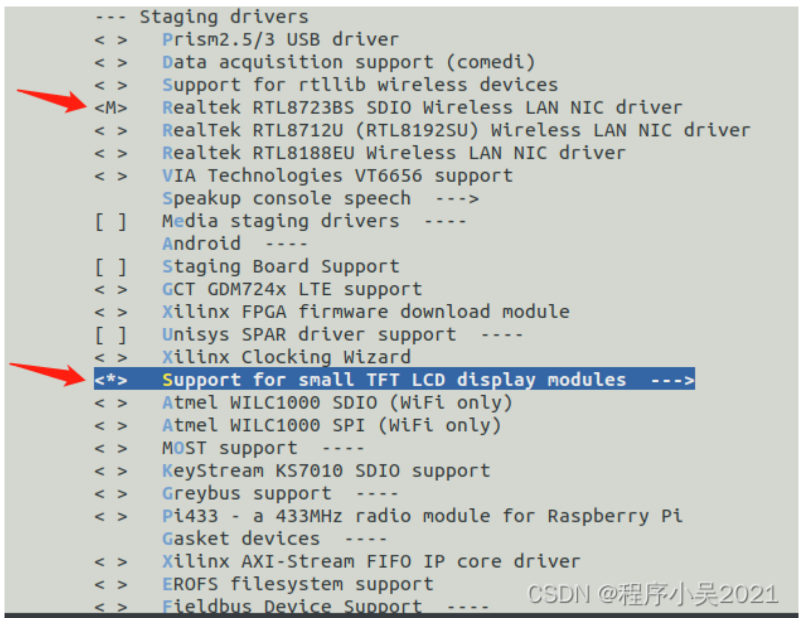
按如下箭头操作



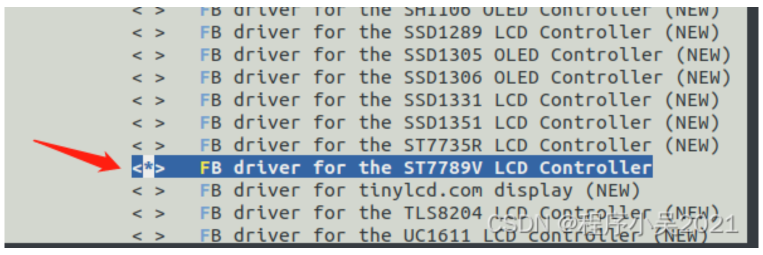
```
onfig - Linux/arm 5.2.0 Kernel Configuration

                    Linux/arm 5.2.0 Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus ---> (or
 Highlighted letters are hotkeys.  Pressing <Y> includes, <N> exc
 features.  Press <Esc><Esc> to exit, <?> for Help, </> for Searc
 [ ] excluded  <M> module  < > module capable
          ^(-)
                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 >
```



```
onfig - Linux/arm 5.2.0 Kernel Configuration
Device Drivers
                              Device Drivers
 Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
 Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
 features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Leg
 [ ] excluded  <M> module  < > module capable
          ^(-)
                Microsoft Hyper-V guest support  ----
                Xen driver support  --->
            [*] Staging drivers  --->
            [ ] Platform support for Goldfish virtual devices  ----
            -*- Platform support for Chrome hardware  --->
            [ ] Platform support for Mellanox hardware (NEW)  ----
                Common Clock Framework  --->
            [*] Hardware Spinlock drivers  ----
                Clock Source drivers  ----
            [*] Mailbox Hardware Support  --->
          (+)

             <Select>    < Exit >    < Help >    < Save >    < Load
```



```
    --- Staging drivers
    < >   Prism2.5/3 USB driver
    < >   Data acquisition support (comedi)
    < >   Support for rtllib wireless devices
    <M>   Realtek RTL8723BS SDIO Wireless LAN NIC driver
    < >   RealTek RTL8712U (RTL8192SU) Wireless LAN NIC driver
    < >   Realtek RTL8188EU Wireless LAN NIC driver
    < >   VIA Technologies VT6656 support
          Speakup console speech  --->
    [ ]   Media staging drivers  ----
          Android  ----
    [ ]   Staging Board Support
    < >   GCT GDM724x LTE support
    < >   Xilinx FPGA firmware download module
    [ ]   Unisys SPAR driver support  ----
    < >   Xilinx Clocking Wizard
    <*>   Support for small TFT LCD display modules  --->
    < >   Atmel WILC1000 SDIO (WiFi only)
    < >   Atmel WILC1000 SPI (WiFi only)
    < >   MOST support  ----
    < >   KeyStream KS7010 SDIO support
    < >   Greybus support  ----
    < >   Pi433 - a 433MHz radio module for Raspberry Pi
          Gasket devices  ----
    < >   Xilinx AXI-Stream FIFO IP core driver
    < >   EROFS filesystem support
    < >   Fieldbus Device Support  ----
```



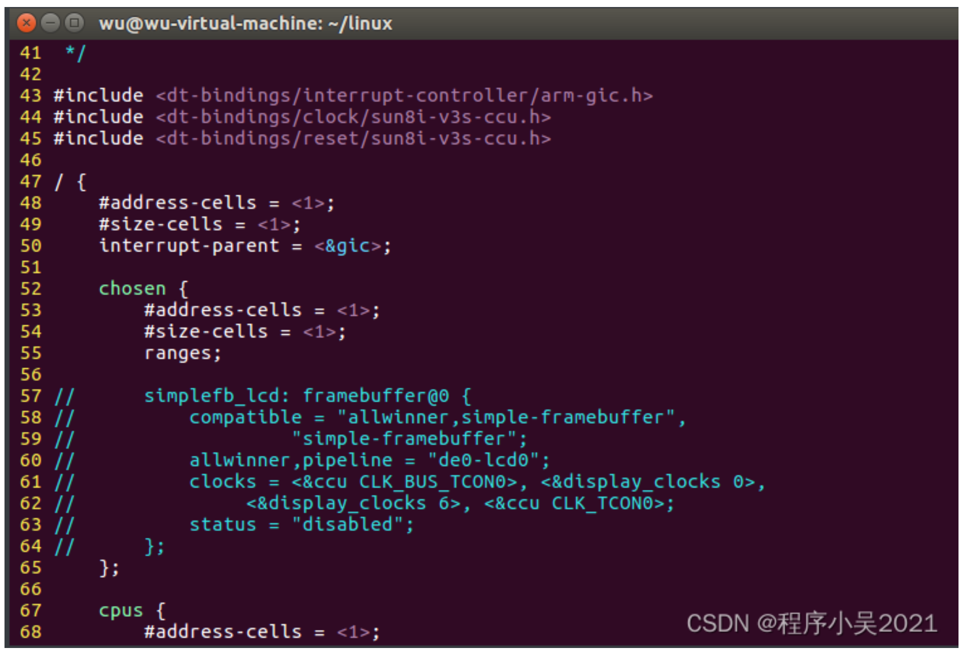
```
    < >   FB driver for the SH1106 OLED Controller (NEW)
    < >   FB driver for the SSD1289 LCD Controller (NEW)
    < >   FB driver for the SSD1305 OLED Controller (NEW)
    < >   FB driver for the SSD1306 OLED Controller (NEW)
    < >   FB driver for the SSD1331 LCD Controller (NEW)
    < >   FB driver for the SSD1351 LCD Controller (NEW)
    < >   FB driver for the ST7735R LCD Controller (NEW)
    <*>   FB driver for the ST7789V LCD Controller
    < >   FB driver for tinylcd.com display (NEW)
    < >   FB driver for the TLS8204 LCD Controller (NEW)
    < >   FB driver for the UC1611 LCD Controller (NEW)
```

然后选Save并退出

要使用lcd要在设备树上做一些修改

1、sun8i-v3s.dtsi中注释掉自带的视频输出

终端输入vim arch/arm/boot/dts/sun8i-v3s.dtsi

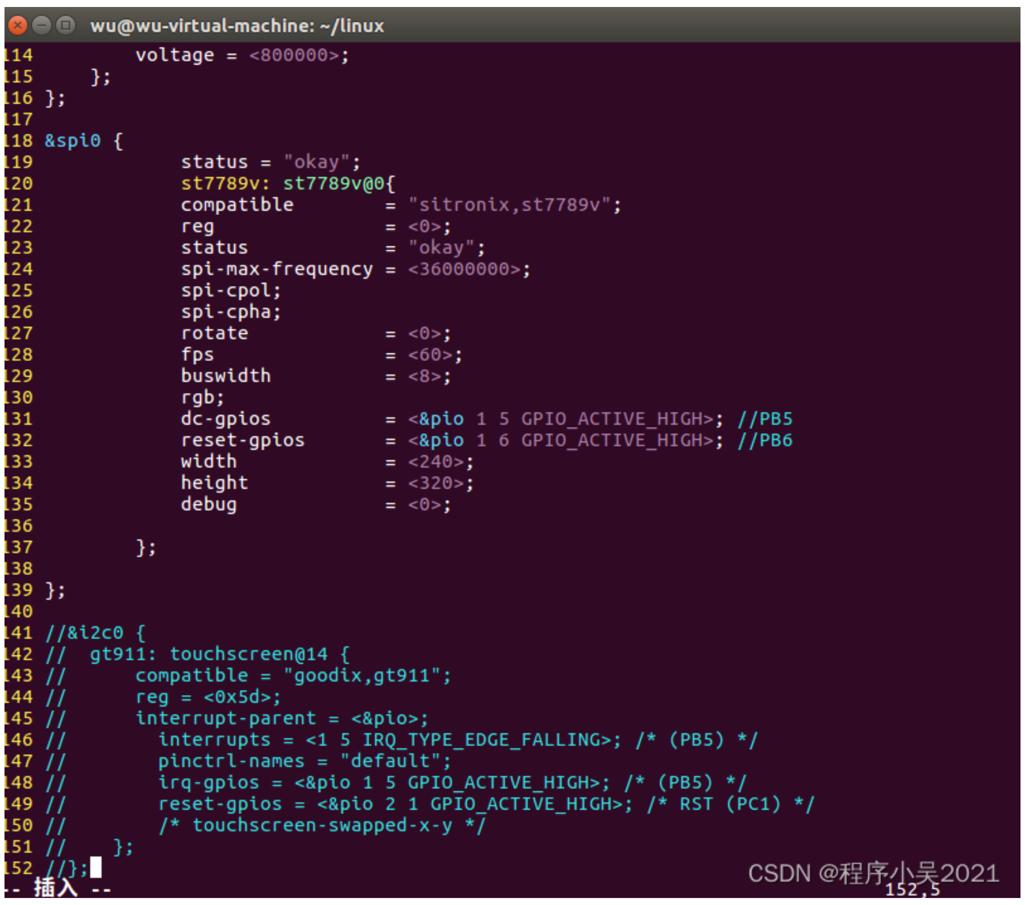


wu@wu-virtual-machine: ~/linux

```
 */

#include <dt-bindings/interrupt-controller/arm-gic.h>
#include <dt-bindings/clock/sun8i-v3s-ccu.h>
#include <dt-bindings/reset/sun8i-v3s-ccu.h>

/ {
	#address-cells = <1>;
	#size-cells = <1>;
	interrupt-parent = <&gic>;

	chosen {
		#address-cells = <1>;
		#size-cells = <1>;
		ranges;

//		simplefb_lcd: framebuffer@0 {
//			compatible = "allwinner,simple-framebuffer",
//				     "simple-framebuffer";
//			allwinner,pipeline = "de0-lcd0";
//			clocks = <&ccu CLK_BUS_TCON0>, <&display_clocks 0>,
//				 <&display_clocks 6>, <&ccu CLK_TCON0>;
//			status = "disabled";
//		};
	};

	cpus {
		#address-cells = <1>;
```

在sun8i-v3s-licheepi-zero-dock.dts增加屏幕驱动节点并注释掉i2c0

```
&spi0 {
status = "okay";
st7789v: st7789v@0{
compatible         = "sitronix,st7789v";
reg                = <0>;
status             = "okay";
spi-max-frequency = <36000000>;
spi-cpol;
spi-cpha;
rotate             = <0>;
fps                = <60>;
buswidth           = <8>;
rgb;
dc-gpios           = <&pio 1 5 GPIO_ACTIVE_HIGH>; //PB5
reset-gpios        = <&pio 1 6 GPIO_ACTIVE_HIGH>; //PB6
```



```
wu@wu-virtual-machine: ~/linux
        voltage = <800000>;
    };
};

&spi0 {
            status = "okay";
            st7789v: st7789v@0{
            compatible          = "sitronix,st7789v";
            reg                 = <0>;
            status              = "okay";
            spi-max-frequency = <36000000>;
            spi-cpol;
            spi-cpha;
            rotate              = <0>;
            fps                 = <60>;
            buswidth            = <8>;
            rgb;
            dc-gpios            = <&pio 1 5 GPIO_ACTIVE_HIGH>; //PB5
            reset-gpios         = <&pio 1 6 GPIO_ACTIVE_HIGH>; //PB6
            width               = <240>;
            height              = <320>;
            debug               = <0>;

        };

};

//&i2c0 {
//  gt911: touchscreen@14 {
//      compatible = "goodix,gt911";
//      reg = <0x5d>;
//      interrupt-parent = <&pio>;
//        interrupts = <1 5 IRQ_TYPE_EDGE_FALLING>; /* (PB5) */
//        pinctrl-names = "default";
//        irq-gpios = <&pio 1 5 GPIO_ACTIVE_HIGH>; /* (PB5) */
//        reset-gpios = <&pio 2 1 GPIO_ACTIVE_HIGH>; /* RST (PC1) */
//        /* touchscreen-swapped-x-y */
//    };
//};
-- 插入 --                                                   152,5
```

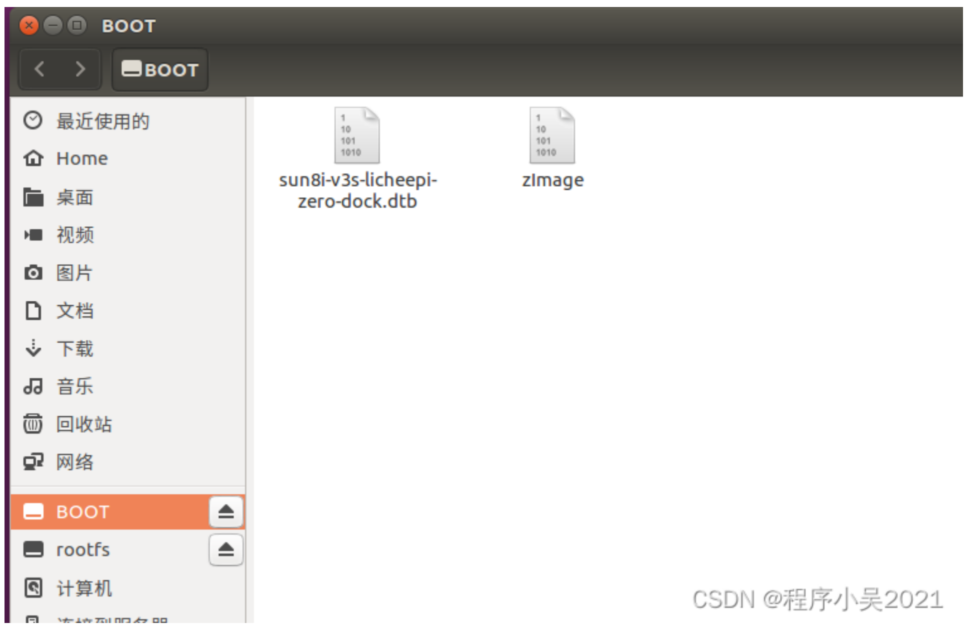
单独编译设备树

终端输入make dtbs

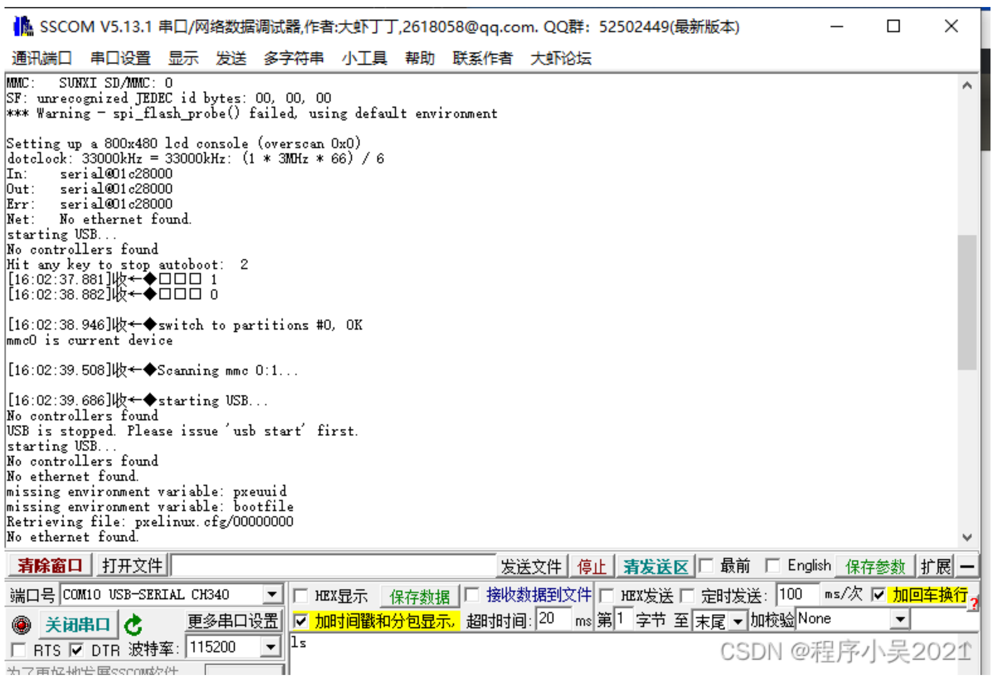
```
make licheepi_zero_defconfig
make menuconfig   #一般不用修改，需要时单独改
make -j16
make  -j16 modules #编译模块
make -j16 modules_install #安装模块
make dtbs #编译设备树
```

编译完成后，zImage在arch/arm/boot/下，驱动模块在out/lib下，设备树在arch/arm/boot/dts下。

然后把zImage，sun8i-v3s-licheepi-zero-dock.dtb放到第一分区



此时插卡上电串口有信息输出，但是没有根文件系统进不去



SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com. QQ群：52502449(最新版本)

```
MMC:   SUNXI SD/MMC: 0
SF: unrecognized JEDEC id bytes: 00, 00, 00
*** Warning - spi_flash_probe() failed, using default environment

Setting up a 800x480 lcd console (overscan 0x0)
dotclock: 33000kHz = 33000kHz: (1 * 3MHz * 66) / 6
In:    serial@01c28000
Out:   serial@01c28000
Err:   serial@01c28000
Net:   No ethernet found.
starting USB...
No controllers found
Hit any key to stop autoboot:  2
[16:02:37.881]收←◆□□□ 1
[16:02:38.882]收←◆□□□ 0

[16:02:38.946]收←◆switch to partitions #0, OK
mmc0 is current device

[16:02:39.508]收←◆Scanning mmc 0:1...

[16:02:39.686]收←◆starting USB...
No controllers found
USB is stopped. Please issue 'usb start' first.
starting USB...
No controllers found
No ethernet found.
missing environment variable: pxeuuid
missing environment variable: bootfile
Retrieving file: pxelinux.cfg/00000000
No ethernet found.
```

端口号 COM10 USB-SERIAL CH340；波特率：115200；[✓ DTR]；[✓ 加时间戳和分包显示]；超时时间：20 ms；第1 字节 至 末尾；加校验 None；定时发送：100 ms/次；[✓ 加回车换行]

```
ls
```

licheepi buildroot根文件系统

首先安装一些依赖，比如linux头文件：

```
apt-get install linux-headers-$(uname -r)
apt-get install libncurses5-dev
apt-get install wget
apt-get install gcc automake autoconf libtool make
```

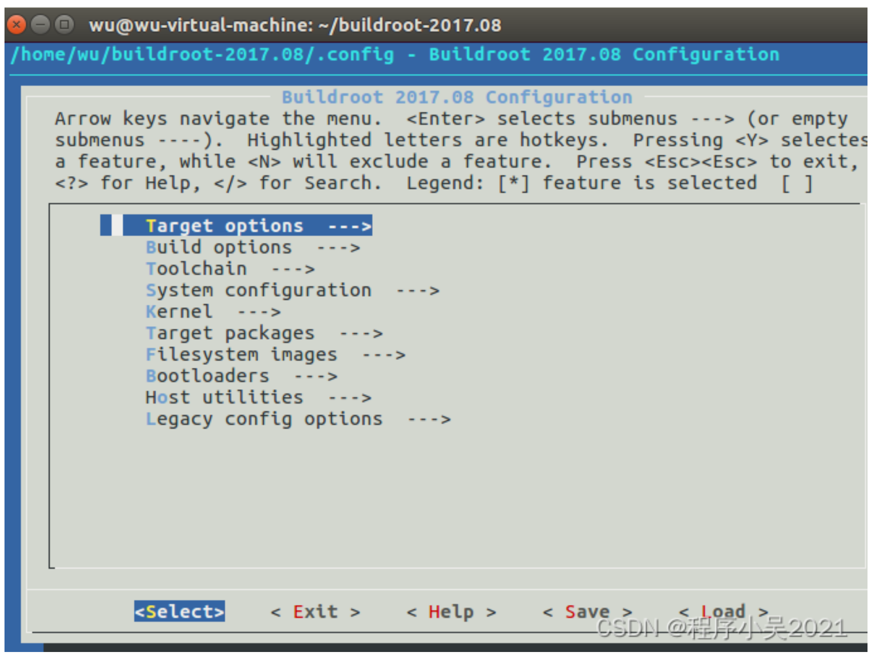
然后下载安装解压：

```
wget https://buildroot.org/downloads/buildroot-2017.08.tar.gz
tar xvf buildroot-2017.08.tar.gz
cd buildroot-2017.08/
make menuconfig
```

ubootroot的目录结构

```
.
├── arch: #存放CPU架构相关的配置脚本，如arm/mips/x86,这些CPU相关的配置，在制作工具链时，编译uboot和kernel时很关键.
├── board
├── boot
├── CHANGES
├── Config.in
├── Config.in.legacy
├── configs: #放置开发板的一些配置参数.
├── COPYING
├── DEVELOPERS
├── dl: #存放下载的源代码及应用软件的压缩包.
├── docs: #存放相关的参考文档.
├── fs: #放各种文件系统的源代码.
├── linux: #存放着Linux kernel的自动构建脚本.
├── Makefile
```

使用 `make config` 进入配置界面。

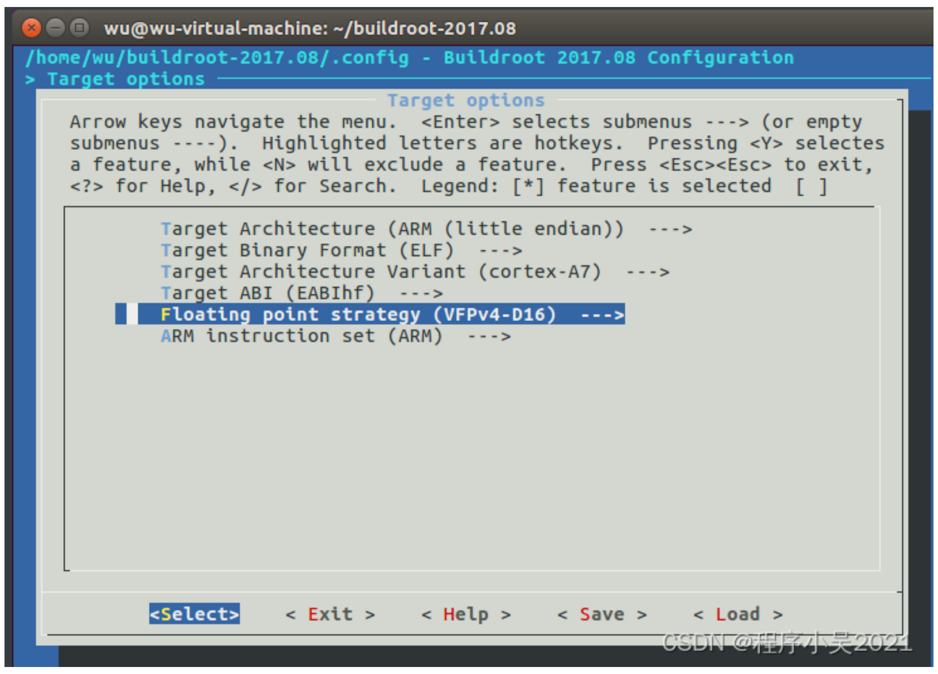


```
wu@wu-virtual-machine: ~/buildroot-2017.08
/home/wu/buildroot-2017.08/.config - Buildroot 2017.08 Configuration
                     Buildroot 2017.08 Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> selectes
  a feature, while <N> will exclude a feature.  Press <Esc><Esc> to exit,
  <?> for Help, </> for Search.  Legend: [*] feature is selected  [ ]

          Target options  --->
          Build options  --->
          Toolchain  --->
          System configuration  --->
          Kernel  --->
          Target packages  --->
          Filesystem images  --->
          Bootloaders  --->
          Host utilities  --->
          Legacy config options  --->

         <Select>    < Exit >    < Help >    < Save >    < Load >
```

选中Target options以选择licheepi对应的架构

licheepi用的v3s cpu 参数如下

```
CPU
ARM Cortex A7 @ 1.2GHz
Support NEON Advanced SIMD instruction
VFPv4 Floating Point Unit
```



```
wu@wu-virtual-machine: ~/buildroot-2017.08
/home/wu/buildroot-2017.08/.config - Buildroot 2017.08 Configuration
> Target options
                          Target options
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> selectes
  a feature, while <N> will exclude a feature.  Press <Esc><Esc> to exit,
  <?> for Help, </> for Search.  Legend: [*] feature is selected  [ ]

          Target Architecture (ARM (little endian))  --->
          Target Binary Format (ELF)  --->
          Target Architecture Variant (cortex-A7)  --->
          Target ABI (EABIhf)  --->
          Floating point strategy (VFPv4-D16)  --->
          ARM instruction set (ARM)  --->

         <Select>    < Exit >    < Help >    < Save >    < Load >
```

Toolchain 配置交叉工具链

配置成外部工具链

```
Toolchain type (External toolchain)  --->
x    ( ) Buildroot toolchain
x    (X) External toolchain

```

在u-boot时已经配置好外部工具，地址在

/opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/

选中 `() Toolchain path (NEW)`，填入path

Toolchain prefix前缀是： **arm-linux-gnueabihf**

External toolchain gcc version：我们使用的是6.3版本,选中6.x

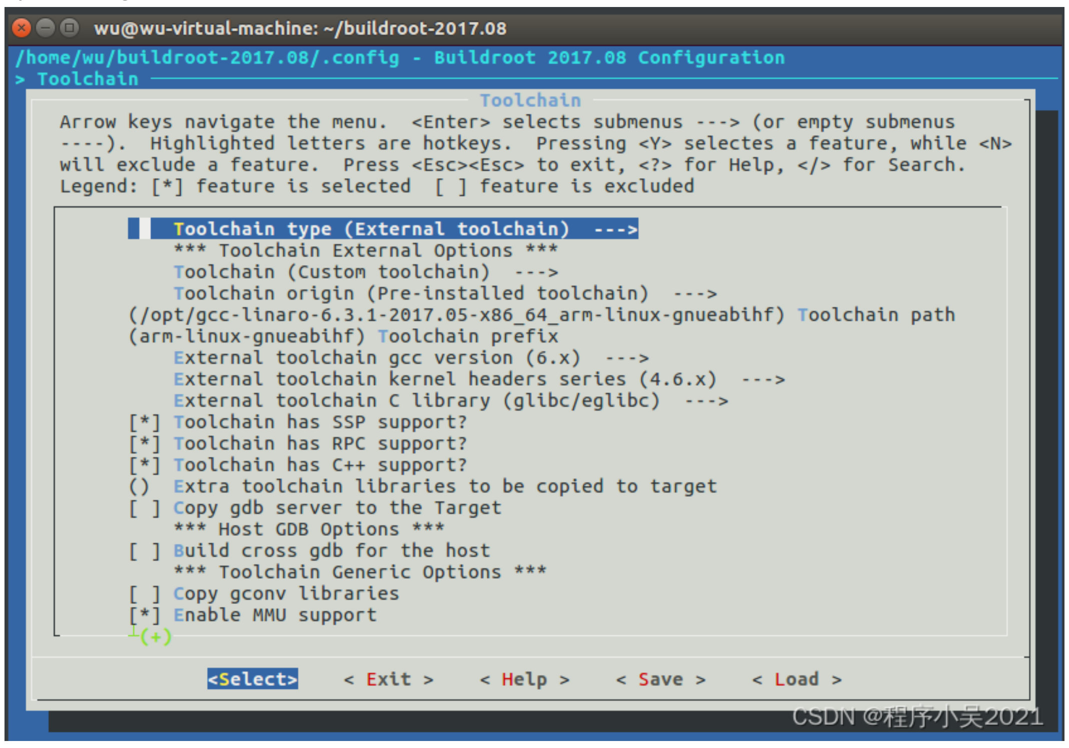
External toolchain kernel headers series：是在
arm-linux-gnueabihf/libc/usr/include/linux/version.h
里读取内核版本信息。本机的版本是4.6(263680=0x040600, 即4.6.0)

```
cat /opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/arm-linux-gnueabihf/libc/usr/include/linux/version.h
#define LINUX_VERSION_CODE 263680

```

External toolchain C library　还是选择传统的glibc。需要小体积可以选uclibc（需要自行编译安装）。然后勾选上 `[] Toolchain has C++ support?`

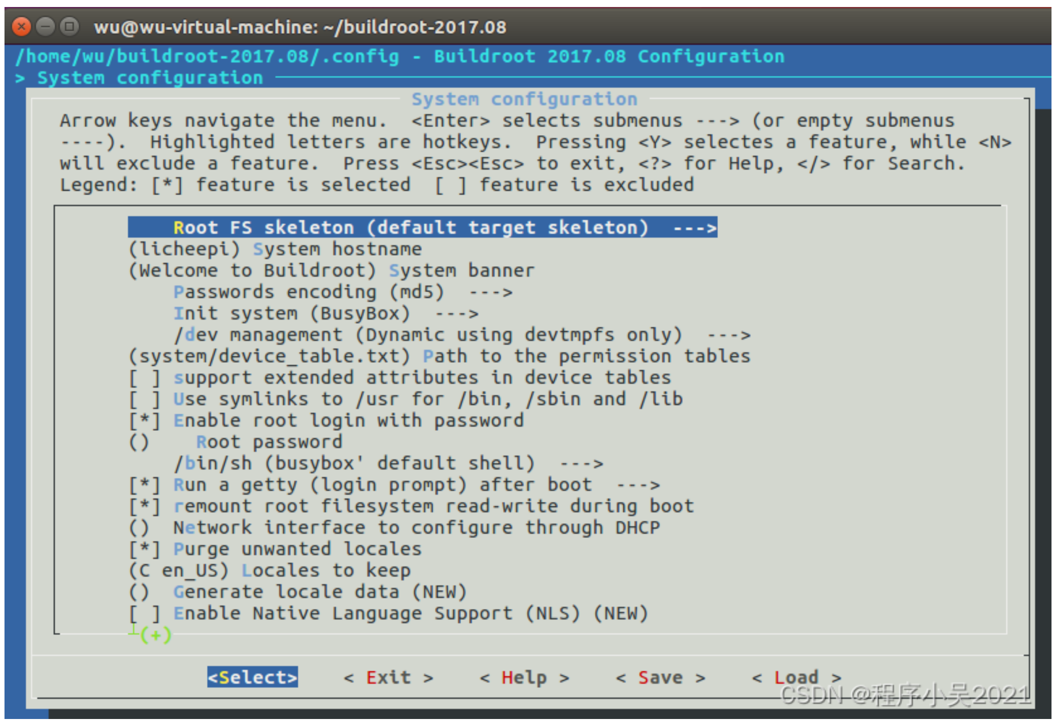
最终配置

System configuration 配置系统参数



```
wu@wu-virtual-machine: ~/buildroot-2017.08
/home/wu/buildroot-2017.08/.config - Buildroot 2017.08 Configuration
> Toolchain
                              Toolchain
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> selectes a feature, while <N>
  will exclude a feature.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] feature is selected  [ ] feature is excluded

        Toolchain type (External toolchain)  --->
        *** Toolchain External Options ***
        Toolchain (Custom toolchain)  --->
        Toolchain origin (Pre-installed toolchain)  --->
    (/opt/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf) Toolchain path
    (arm-linux-gnueabihf) Toolchain prefix
        External toolchain gcc version (6.x)  --->
        External toolchain kernel headers series (4.6.x)  --->
        External toolchain C library (glibc/eglibc)  --->
    [*] Toolchain has SSP support?
    [*] Toolchain has RPC support?
    [*] Toolchain has C++ support?
    ()  Extra toolchain libraries to be copied to target
    [ ] Copy gdb server to the Target
        *** Host GDB Options ***
    [ ] Build cross gdb for the host
        *** Toolchain Generic Options ***
    [ ] Copy gconv libraries
    [*] Enable MMU support
    ┴(+)

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

- System hostname：根据需要定义一个字符串，是控制台前面的提示符xxx@vsi，这里改为 `(licheepi) System hostname`
- Init system：这里选择busybox，轻量级使用非常广泛。可选的有systemV,systemd.
- Root password配置登录密码。



```
wu@wu-virtual-machine: ~/buildroot-2017.08
/home/wu/buildroot-2017.08/.config - Buildroot 2017.08 Configuration
> System configuration
                           System configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> selectes a feature, while <N>
  will exclude a feature.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] feature is selected  [ ] feature is excluded

        Root FS skeleton (default target skeleton)  --->
    (licheepi) System hostname
    (Welcome to Buildroot) System banner
        Passwords encoding (md5)  --->
        Init system (BusyBox)  --->
        /dev management (Dynamic using devtmpfs only)  --->
    (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
    ()      Root password
        /bin/sh (busybox' default shell)  --->
    [*] Run a getty (login prompt) after boot  --->
    [*] remount root filesystem read-write during boot
    ()  Network interface to configure through DHCP
    [*] Purge unwanted locales
    (C en_US) Locales to keep
    ()  Generate locale data (NEW)
    [ ] Enable Native Language Support (NLS) (NEW)
    ┴(+)

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

Target package

用于配置一些软件包，例如QT5

```
Target packages  --->
x  Graphic libraries and applications (graphic/text)  --->
xx   [*] Qt5  --->
```

配置完成退出保存，安装环境

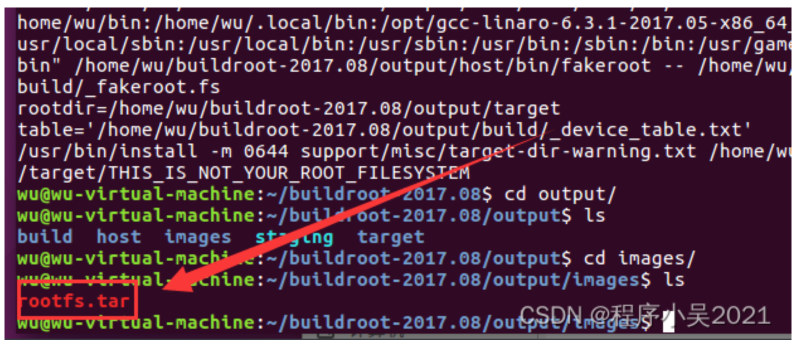
sudo apt-get install g++ patch cpio python unzip rsync bc

编译

make

如果报错，make clean all一下重新编译，这里一定要注意gcc的路径，不然会过不去

编译完成后会在output/images下生成rootfs.tar



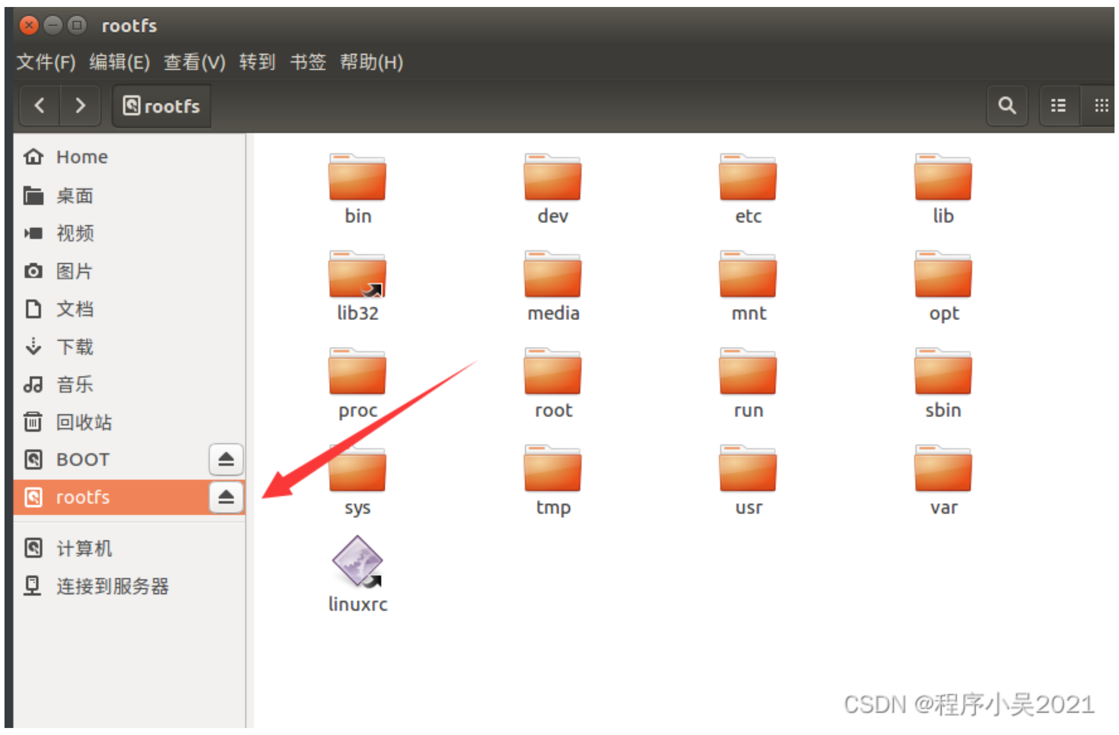
```
home/wu/bin:/home/wu/.local/bin:/opt/gcc-linaro-6.3.1-2017.05-x86_64
usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/gam
bin" /home/wu/buildroot-2017.08/output/host/bin/fakeroot -- /home/wu
build/_fakeroot.fs
rootdir=/home/wu/buildroot-2017.08/output/target
table='/home/wu/buildroot-2017.08/output/build/_device_table.txt'
/usr/bin/install -m 0644 support/misc/target-dir-warning.txt /home/w
/target/THIS_IS_NOT_YOUR_ROOT_FILESYSTEM
wu@wu-virtual-machine:~/buildroot-2017.08$ cd output/
wu@wu-virtual-machine:~/buildroot-2017.08/output$ ls
build  host  images  staging  target
wu@wu-virtual-machine:~/buildroot-2017.08/output$ cd images/
wu@wu-virtual-machine:~/buildroot-2017.08/output/images$ ls
rootfs.tar
wu@wu-virtual-machine:~/buildroot-2017.08/output/images$
```

解压到第二分区后就能使用了

我喜欢直接拖动文件，启动超级文件权限

sudo nautilus

将子文件全部拖出来，不然还是会进不去系统

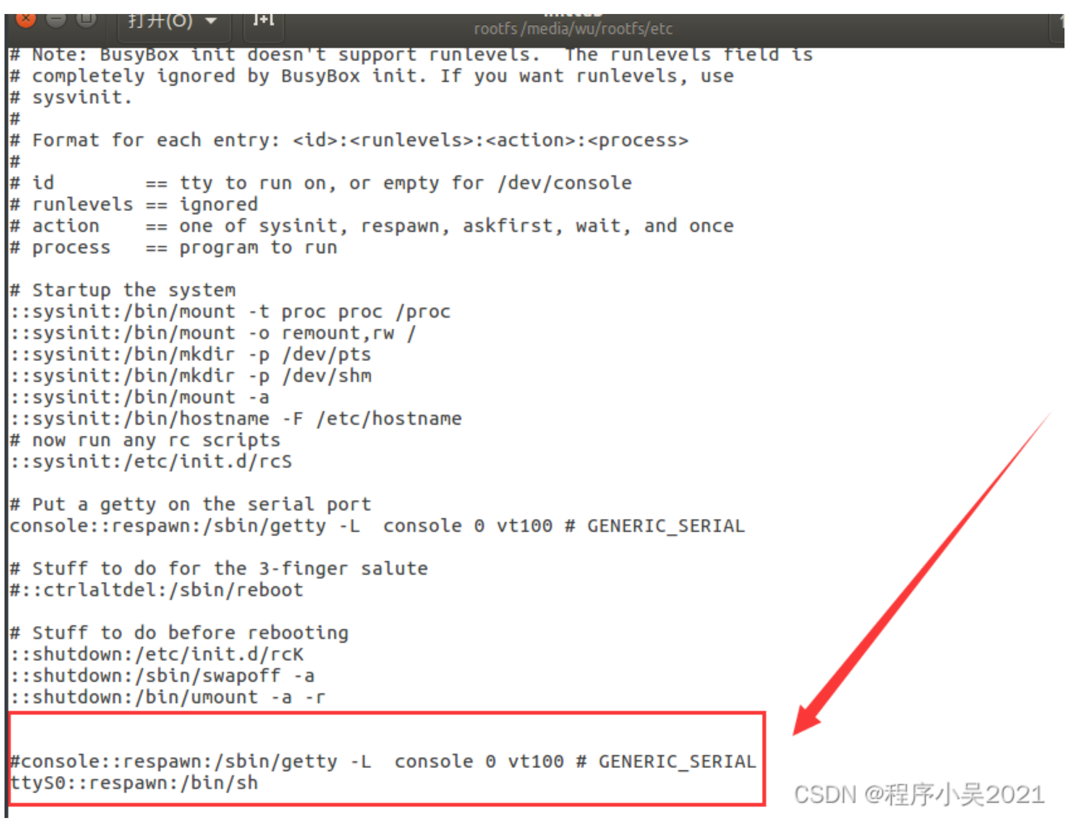


默认失能串口登录，需要修改 **/etc/inittab**：

```
#console::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL
ttyS0::respawn:/sbin/getty -L ttyS0 115200 vt100 # GE

```

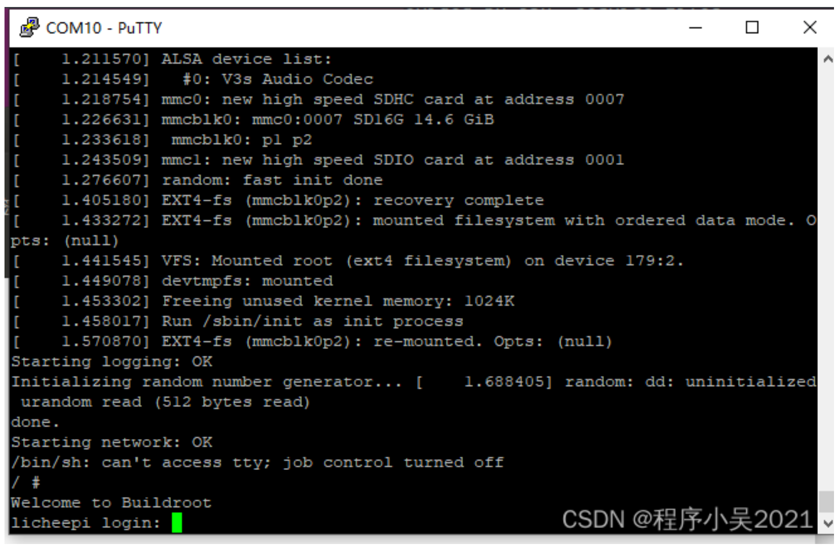
如果需要免密码登录，直接

```
#console::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL
ttyS0::respawn:/bin/sh


```



```
# Note: BusyBox init doesn't support runlevels.  The runlevels field is
# completely ignored by BusyBox init. If you want runlevels, use
# sysvinit.
#
# Format for each entry: <id>:<runlevels>:<action>:<process>
#
# id        == tty to run on, or empty for /dev/console
# runlevels == ignored
# action    == one of sysinit, respawn, askfirst, wait, and once
# process   == program to run

# Startup the system
::sysinit:/bin/mount -t proc proc /proc
::sysinit:/bin/mount -o remount,rw /
::sysinit:/bin/mkdir -p /dev/pts
::sysinit:/bin/mkdir -p /dev/shm
::sysinit:/bin/mount -a
::sysinit:/bin/hostname -F /etc/hostname
# now run any rc scripts
::sysinit:/etc/init.d/rcS

# Put a getty on the serial port
console::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#::ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
::shutdown:/etc/init.d/rcK
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r

#console::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL
ttyS0::respawn:/bin/sh
```

插卡，完成



```
[    1.211570] ALSA device list:
[    1.214549]   #0: V3s Audio Codec
[    1.218754] mmc0: new high speed SDHC card at address 0007
[    1.226631] mmcblk0: mmc0:0007 SD16G 14.6 GiB
[    1.233618]  mmcblk0: p1 p2
[    1.243509] mmc1: new high speed SDIO card at address 0001
[    1.276607] random: fast init done
[    1.405180] EXT4-fs (mmcblk0p2): recovery complete
[    1.433272] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. O
pts: (null)
[    1.441545] VFS: Mounted root (ext4 filesystem) on device 179:2.
[    1.449078] devtmpfs: mounted
[    1.453302] Freeing unused kernel memory: 1024K
[    1.458017] Run /sbin/init as init process
[    1.570870] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
Starting logging: OK
Initializing random number generator... [    1.688405] random: dd: uninitialized
 urandom read (512 bytes read)
done.
Starting network: OK
/bin/sh: can't access tty; job control turned off
/ #
Welcome to Buildroot
licheepi login: 
```