AHL-STM32MP157 快速指南-20210317

第 1 章 概述

AHL-STM32MP157 是集嵌入式 Linux 系统应用与驱动程序的编辑、编译以及运行为一体的开发体系。其旨在于简化嵌入式 Linux 应用与驱动程序从编辑到编译到运行的开发过程中需要经历的复杂人机交互，利用尽可能少的开发工具实现对嵌入式 Linux 应用与驱动程序的快速开发与运行。本体系具有编辑应用与驱动程序、多渠道编译应用与驱动程序、多渠道更新设备树、多渠道运行应用与驱动程序等功能，其丰富的应用场景以及普适性的特点，为开发人员提供了一个方便快捷的嵌入式 Linux 开发平台。

第 2 章 硬件快速测试

2.1 硬件资源

本体系所使用的开发板名为 AHL-STM32MP157 开发板，其中核心板部分使用的是米尔科技研发的基于 STM32MP157 芯片的 MYC-YA157C 核心板，扩展板部分由苏州大学嵌入式人工智能与物联网实验室自主研发，开发板实物图如图 1 所示。

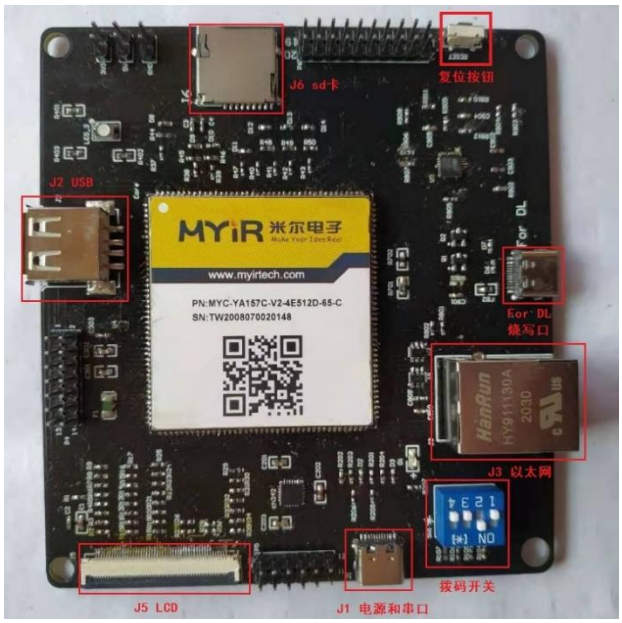

图 1 AHL-STM32MP157 开发板实物图

其中扩展板外设接口列表说明如表 1 所示。

表 1 扩展板外设接口列表

外设名称	说明
J1 电源和串口type-C接口	提供开发板电源、Debug串口以及用户串口
拨码开关	用于开发板启动模式选择
J2 USB	通用外部USB设备接口
J3 以太网	有线网卡接口

For DL	嵌入式Linux操作系统烧写口
J5 LCD	LCD显示接口
J6 sd卡	sd卡接口
复位按钮	用于开发板热复位

2.2 硬件测试

将已烧录进嵌入式 Linux 操作系统的开发板通过 J1 口连接电脑 USB 口供电，打开 06_Tool 中的串口工具 XCOM.exe，选择 Debug 串口对应的串口号（因为该接口同时提供了 Debug 串口和用户串口，故打开时可通过打开两个 XCOM 串口工具，将两个串口号都打开以便找到 Debug 串口对应的串口号），可观察到有开发板启动过程中打印出的输出内容，最后启动成功后应如图 2 所示（串口号可能不同）。

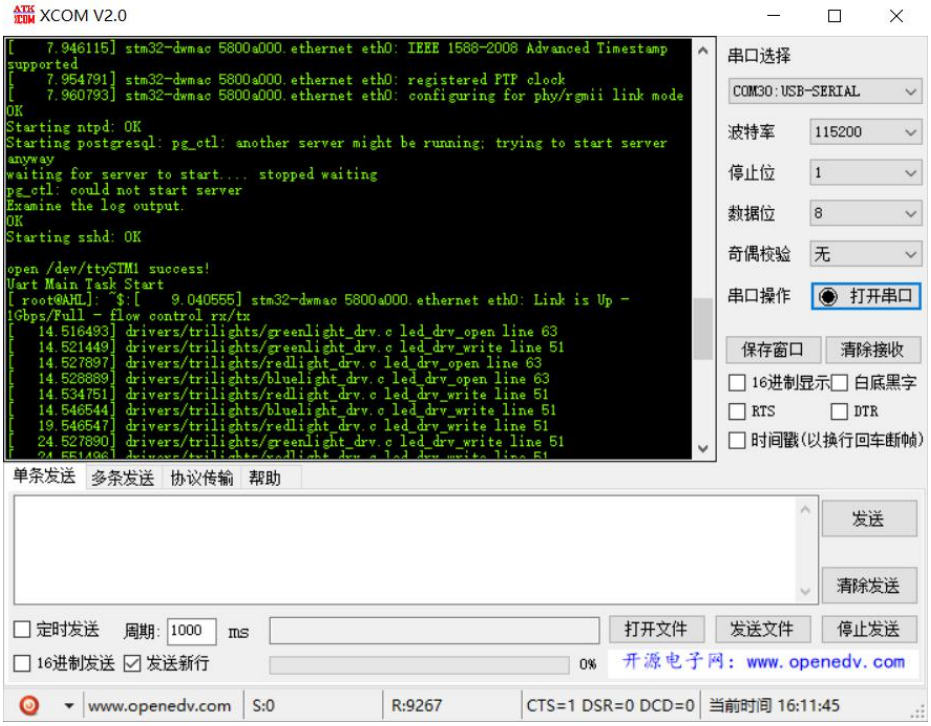

图 2 开发板启动成功输出

同时可观察到开发板上三色灯闪烁，闪烁颜色循环顺序为：暗、红、绿、黄、蓝、紫、青、白。

若出现上述现象，说明硬件资源完好。

第 3 章 开发环境搭建与测试程序运行

3.1 下载电子资源（已包含在拷贝的资料里）

本快速指南所涉及的开发体系软件资源均可至苏州大学嵌入式学习社区 sumcu.suda.edu.cn 下的“金葫芦专区”→“AHL-STM32MP157”，选择“AHL-STM32MP157-20210221”下载，如图 3 所示。

图 3 电子资源下载地址

电子资源包含内容说明如表 2 所示。

表 2 电子资源包含内容

名称	说明
01_Doc	AHL-STM32MP157相关说明文档，包含STM32MP157芯片的参考手册、数据手册以及架构文档等
02_Soft	包含集成开发环境软件ELinux-IDE的安装包以及测试程序first_prg
03_Hard	包含AHL-STM32MP157开发板电路原理图
04_Image	包含嵌入式Linux操作系统烧写镜像
05_Source	包含搭建开发环境时所需的资源文件
06_Tool	包含串口工具XCOM.exe

3.2 开发环境搭建

电子资源下载完毕后，应当进行必要的开发环境搭建，以保证编译与运行时网络通信等方式的正常运行。该开发环境搭建方法仅适用于 Windows 10 build 16299 及其以上版本。其他版本请参考附录 1 进行虚拟机上的 Linux 编译环境搭建。

1．开启 Windows 上的 Linux 功能

打开电脑控制面板→卸载程序→启用或关闭 Windows 功能，找到“适用于 Linux 的 Windows 子系统”，点击勾选，如图 4 所示。这一步有可能需要重启。

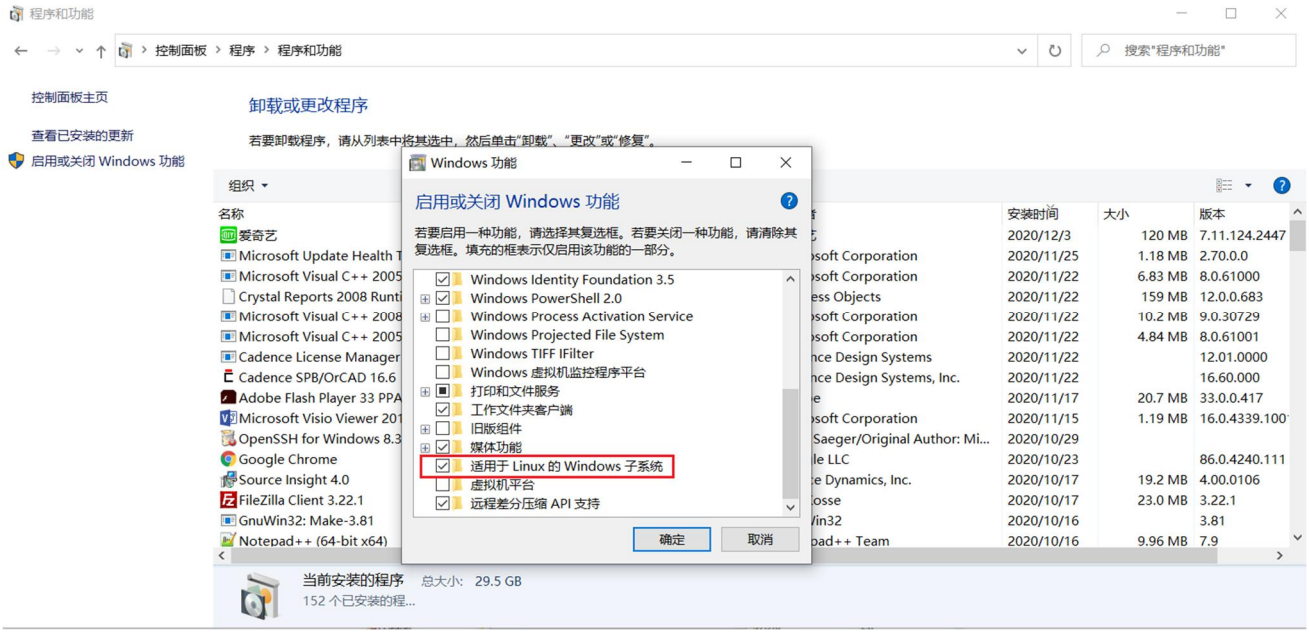

图 4 开启 Windows 上的 Linux 功能

2．安装 Ubuntu18.04

打开 Microsoft Store，搜索 Ubuntu18.04，点击“Install”安装，如图 5 所示。

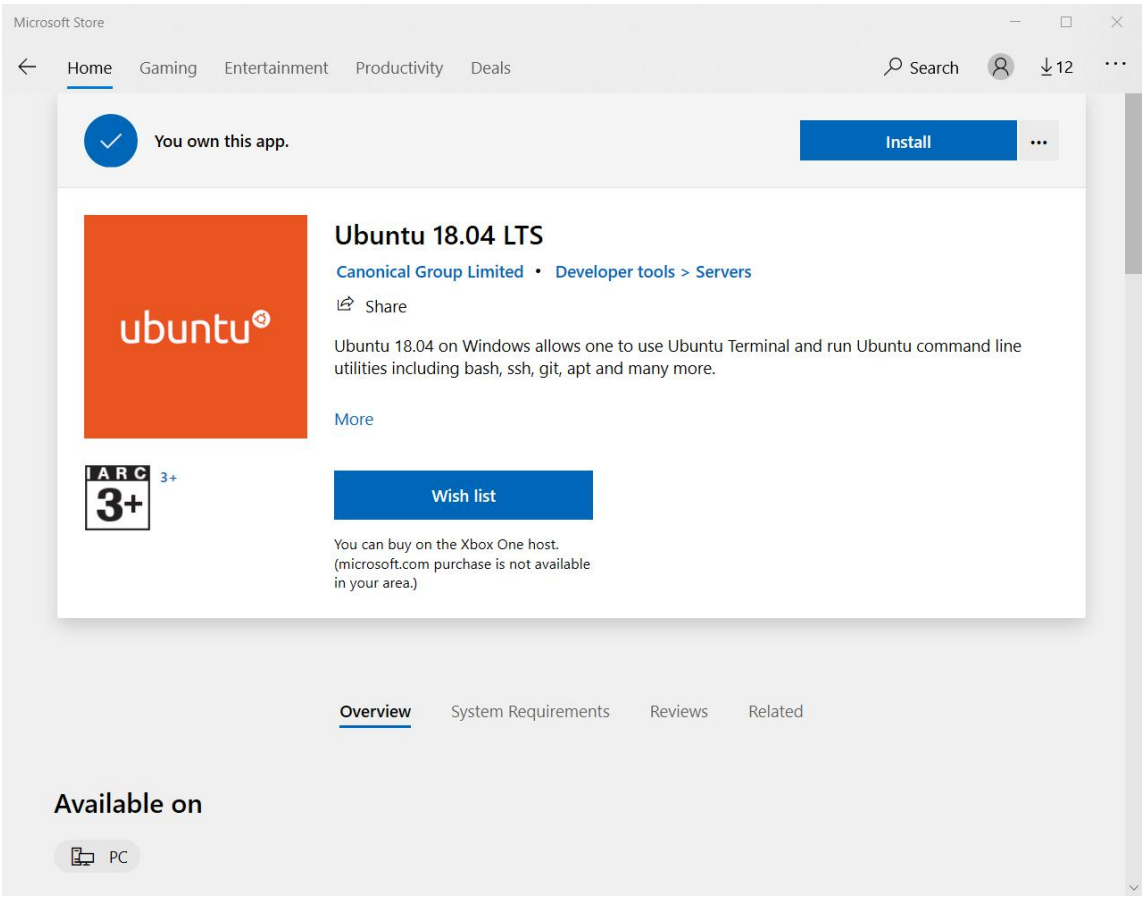

图 5 安装 Ubuntu18.04

安装完成后，点击“Launch”按钮打开，会弹出终端，显示正在安装 Ubuntu，等待安装成功后，输入用户名和密码，如图 6 所示。

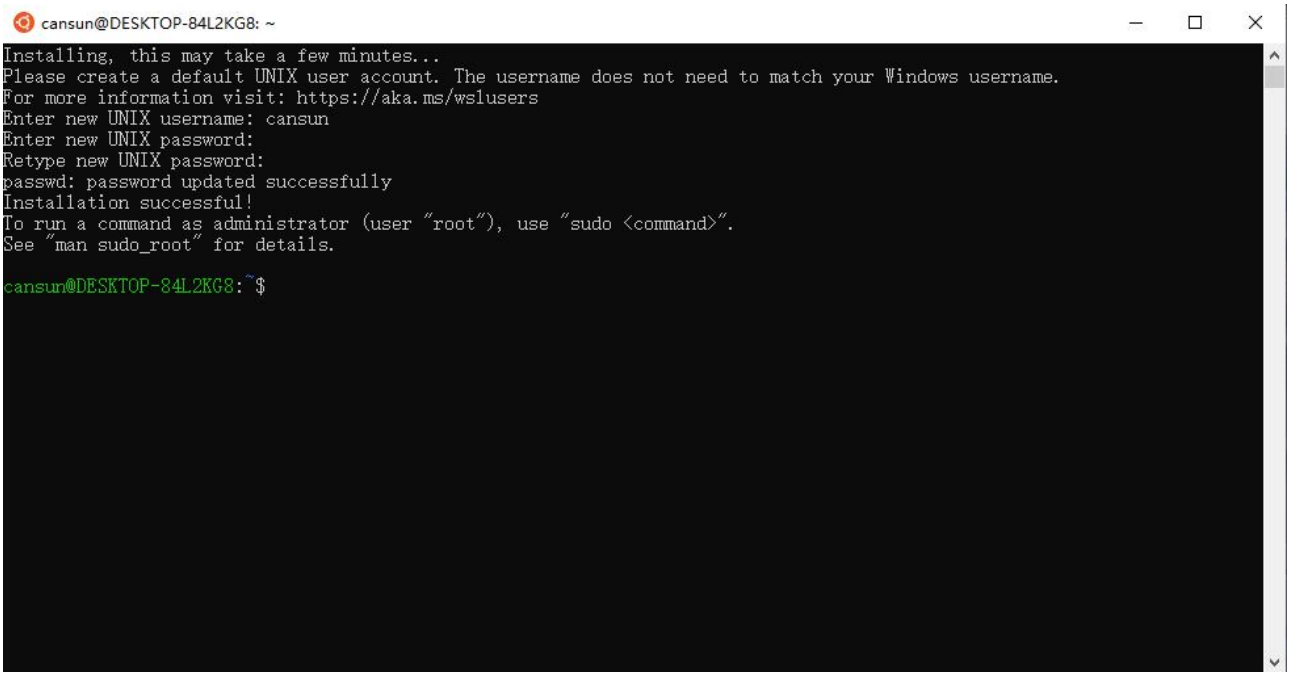

图 6 显示安装界面

3．开启 WindowsApp 文件夹执行权限

打开“我的电脑”，进入 C:\Program Files 目录下，找到 WindowsApps 文件夹（若找不到该文件夹，在文件资源管理器上方选择“查看”，勾选“隐藏的项目”，即可看到该文件夹），右击“属性”→“安全”→“高级”→“更改”，在“输入要选择的对象名称”一栏中输入“Everyone”，如图 7 所示。

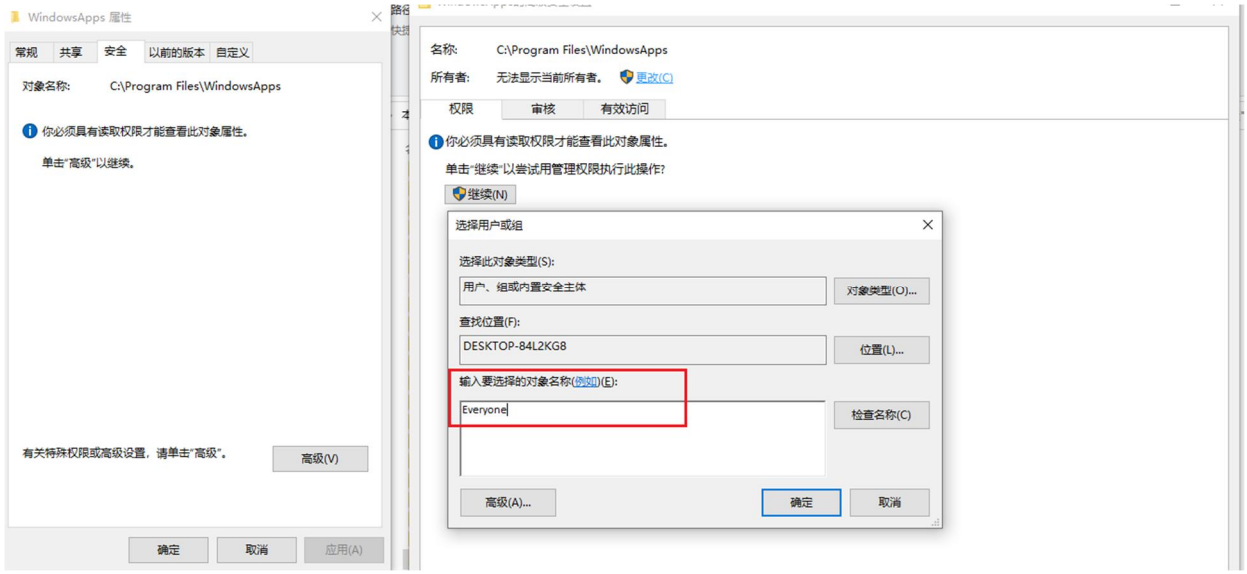

图 7 开启 WindowsApp 文件夹执行权限-1

点击“确定”，然后勾选“替换子容器和对象的所有者”，点击“应用”，如图 8 所示.

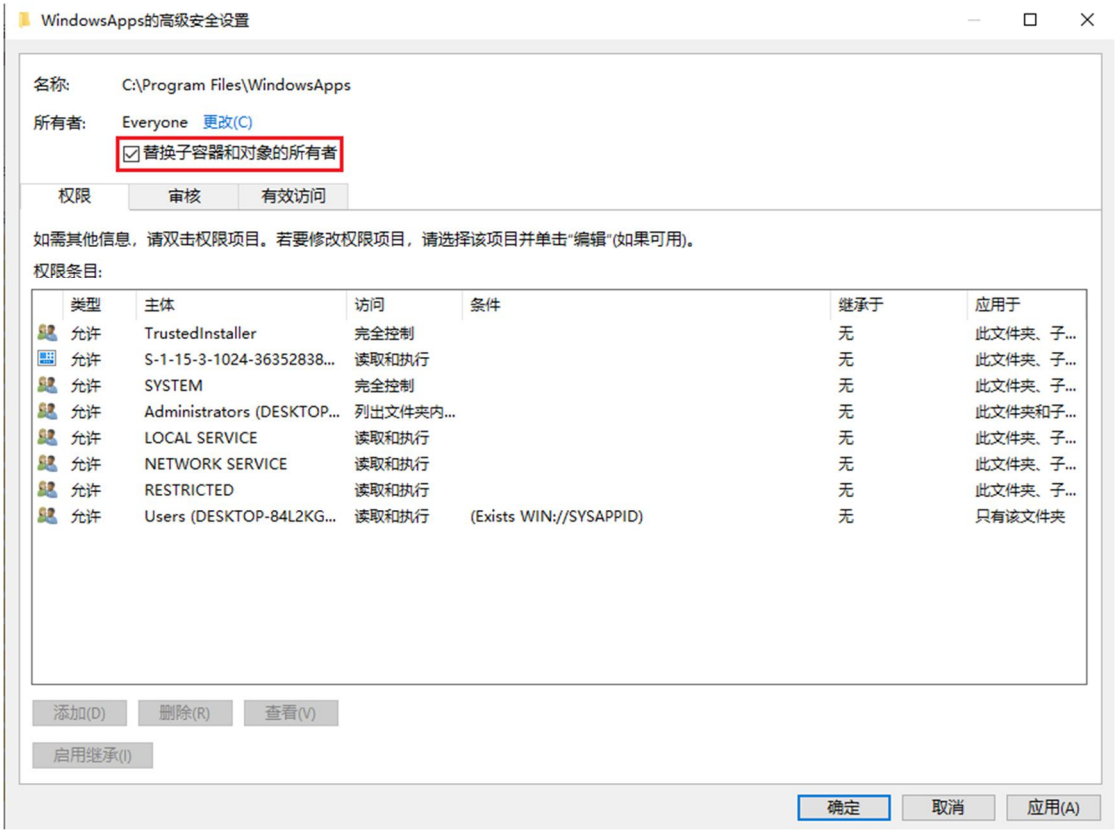

图 8 开启 WindowsApp 文件夹执行权限-2

4．自动化开发环境搭建

运行自动化开发环境搭建软件，初始界面如图 9 所示。

图 9 自动化开发环境搭建软件初始界面

1）设置 Ubuntu 默认用户为 root

点击 “设置 Ubuntu 默认用户为 root” 按钮，会弹出更改成功提示框，如图 10 所示。

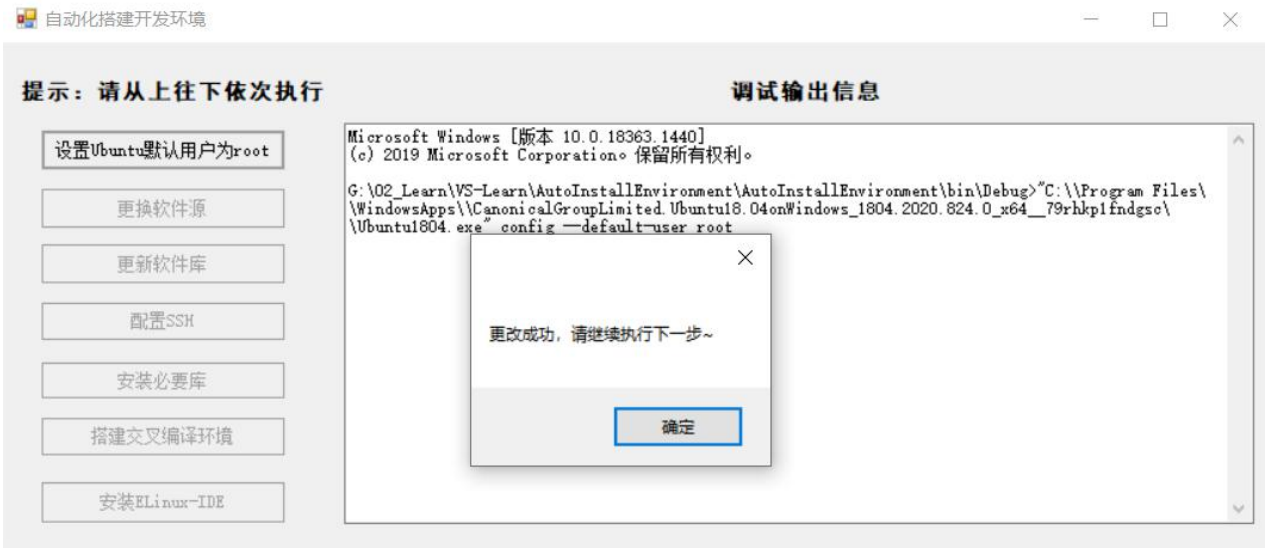

图 10 设置 Ubuntu 默认用户为 root

2）更换软件源

点击“更换软件源”按钮，耐心等待，会弹出执行完成提示框，如图 11 所示。

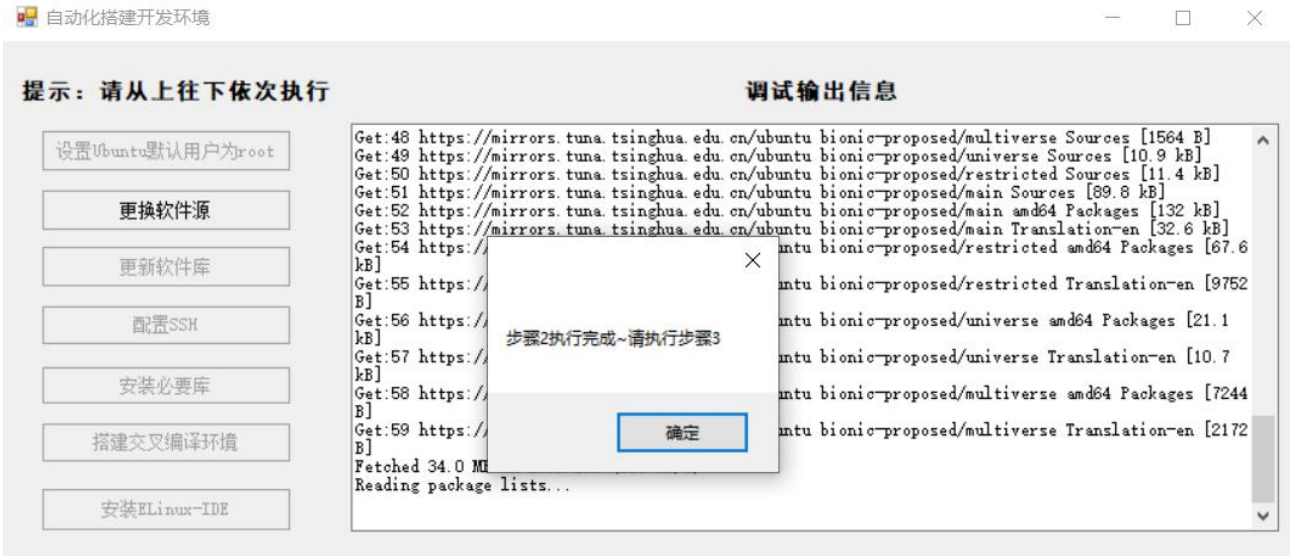

图 11 更换软件源

3）更换软件库

点击“更换软件库”按钮，本步骤与电脑网速有关，执行时间较长，耐心等待执行完成，会弹出执行完成提示框，如图 12 所示。

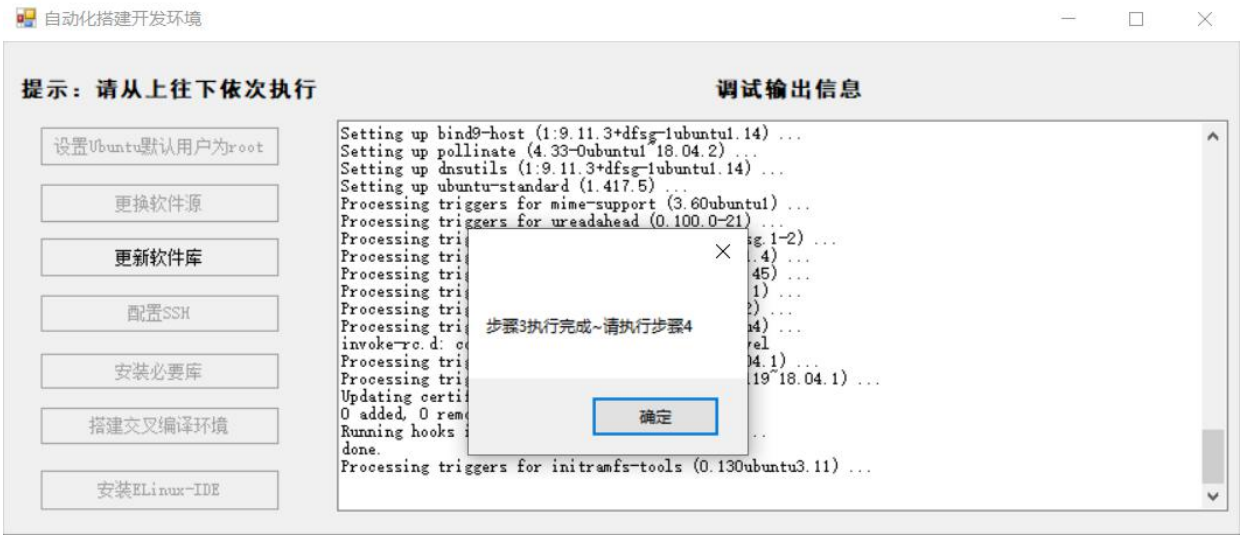

图 12 更新软件库

4）配置 SSH

点击“配置 SSH”按钮，会弹出执行完成提示框，如图 13 所示。

图 13 配置 SSH

5）安装必要库

点击“安装必要库”按钮，安装编译时需要用的 Linux 库，耐心等待执行完成，会弹出执行完成提示框，如图 14 所示。

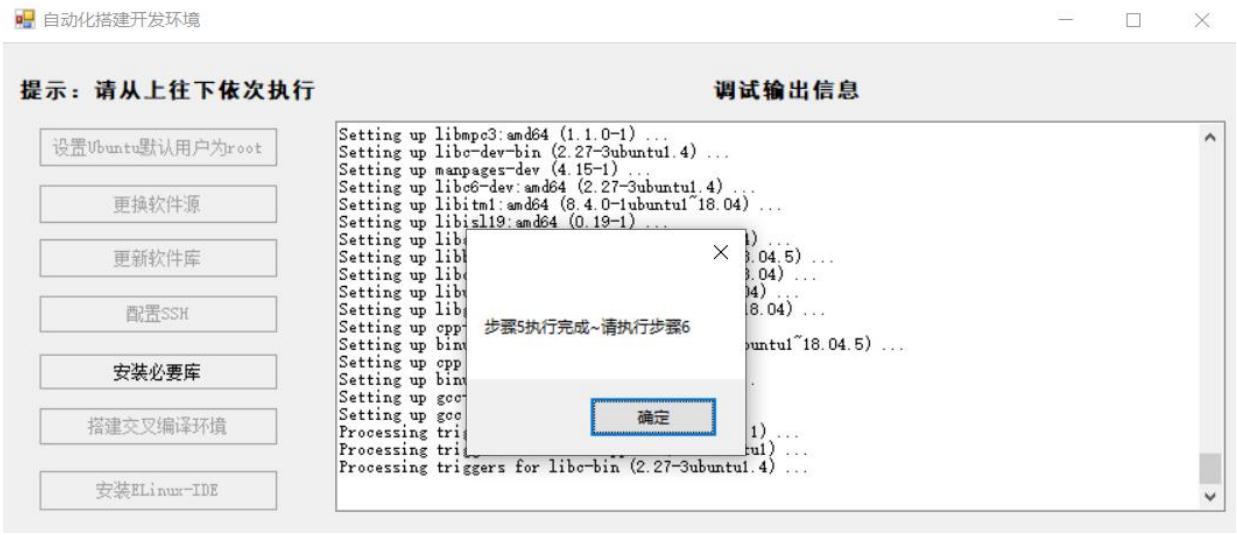

图 14 安装必要库

6）搭建交叉编译环境

点击“搭建交叉编译环境”按钮，耐心等待执行完成，会弹出执行完成提示框，如图 15 所示。

图 15 搭建交叉编译环境

7）安装 ELinux-IDE

点击“安装 ELinux-IDE”按钮，进行 ELinux-IDE 的安装，注意执行此步骤之前需要关闭电脑防火墙和杀毒软件，否则会出现错误！

安装完成后重启电脑，使自动化开发环境搭建生效。

5．共享无线网卡

将网线的一端连接至电脑，一端连接至开发板，启动开发板，在电脑右下角的 wifi 标志右击，选择“打开网络和 Internet 设置”→“更改适配器选项”。找到刚才插入网线时显示出来的以太网，例如我的是“以太网 4”，如图 16 所示。

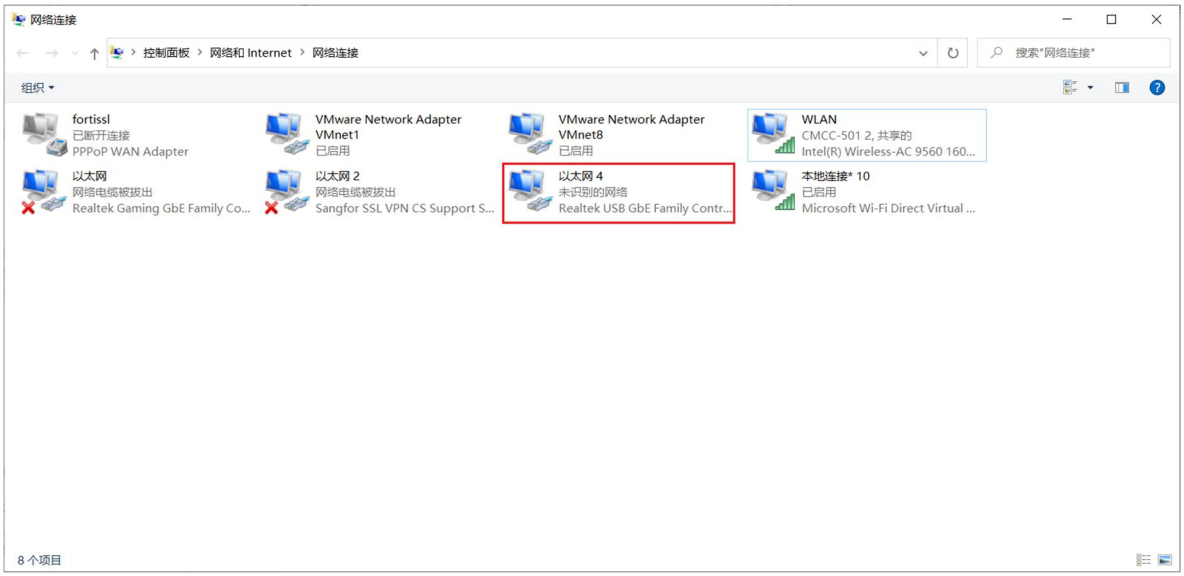

图 16 找到要修改的以太网

找到无线网卡，例如我的是“WLAN”，右击选择“属性”→“共享”，将“允许其他网络用户通过此计算机的 Internet 连接来连接”勾上，在 “家庭网络连接”一栏输入“以太网 4”，如图 17 所示。

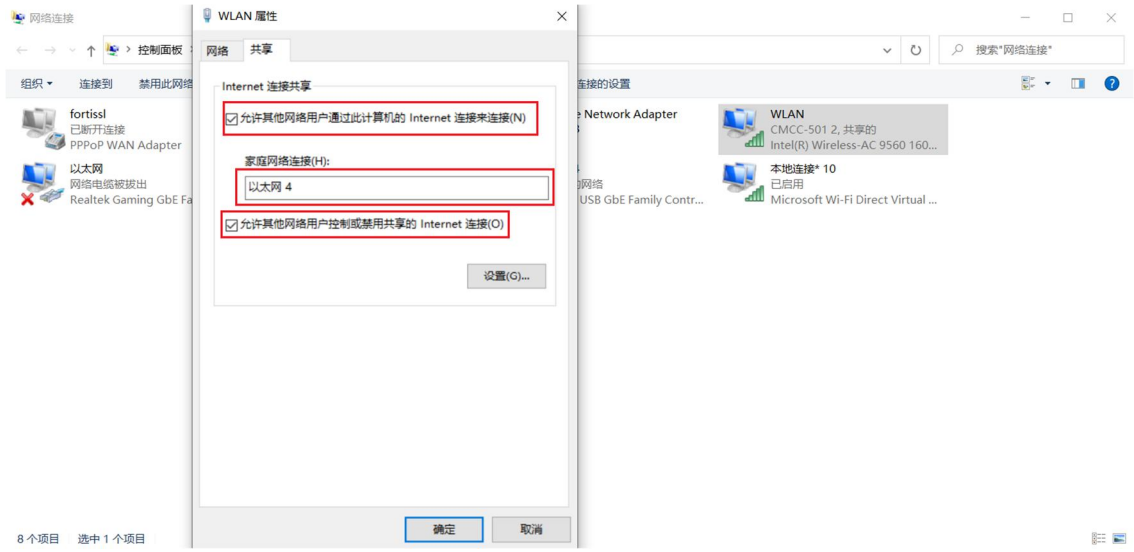

图 17 共享无线网卡

注意：当切换其他无线网络时，该步需要重新配置。

6. 配置本地网络连接静态 IP

右击步骤 5 中选择的以太网，选择“属性”→双击“Internet 协议版本 4（TCP/IPv4）”，配置静态 IP 地址，如图 18 所示。

图 18 配置本地连接静态 IP

至此开发环境搭建完成。

3.3 编译、下载第一个程序

1．导入工程

运行安装好的 ELinux-IDE[①]软件，单击导入工程，打开 02_Soft 下的示例工程“first_prg-20210128”，如图 19 所示。

① 该软件的具体功能介绍可参见附录 2 ELinux-IDE 使用指南

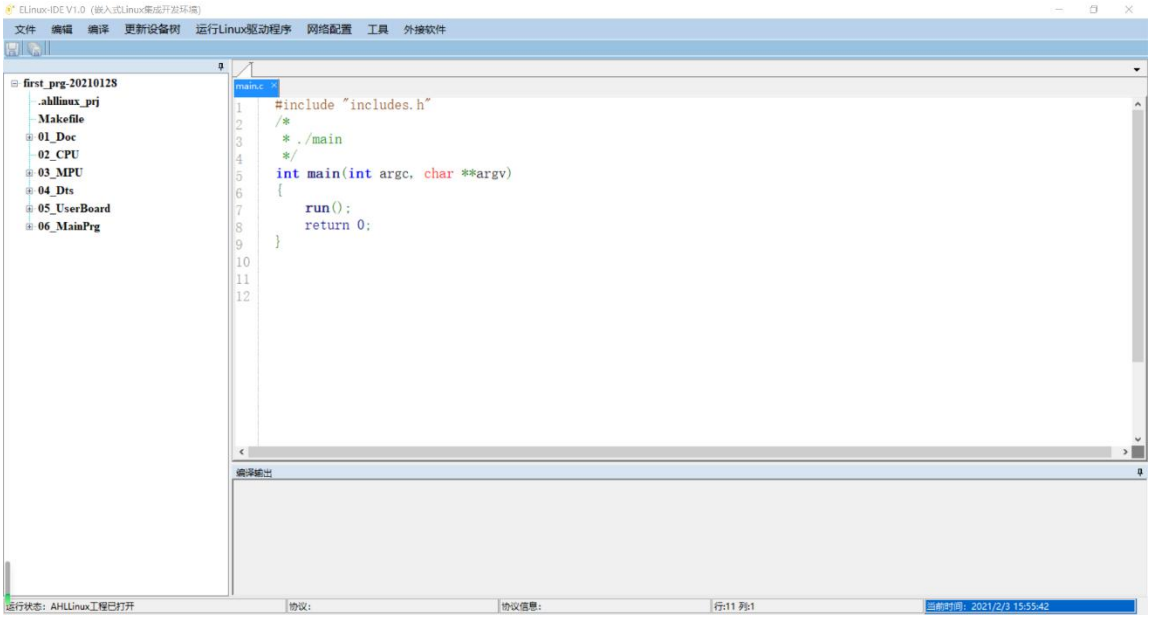

图 19 ELinux-IDE 打开示例工程

2．配置网络

点击 ELinux-IDE 上方的“网络配置”按钮，在弹出的框中根据自己之前获取的内容，配置开发板 IP（默认为 192.168.31.100）、开发板用户名（默认为 root）、开发板密码（默认为 123456），如图 20 所示。如果在开发环境搭建时编译环境使用的是虚拟机 Ubuntu 的方式，则还需要配置虚拟机 Linux 的 IP、用户名以及密码。

图 20 配置网络

配置完成后，点击“确认”即可。

3．更新设备树

打开本地 Ubuntu。点击 ELinux-IDE 上方的“更新设备树→本地 Linux 更新”按钮，会自动进行设备树

更新，等待弹窗提示“更新设备树成功，请重启开发板~”如图 21 所示，按开发板上复位键进行开发板重启等待其开机完成（三色灯亮）即可。

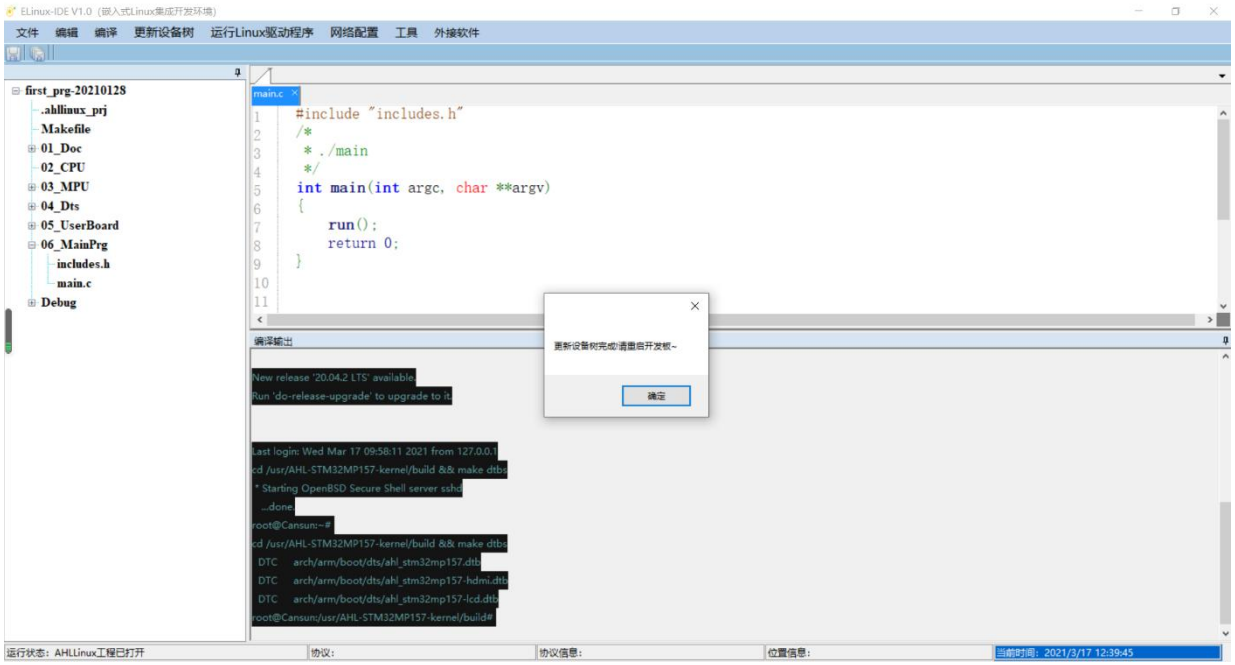

图 21 更新设备树

4．编译程序

点击 ELinux-IDE 上方的“编译→本地编译 Linux 驱动”按钮，下方控制台会输出编译过程。编译完成后会在工程目录下生成 Debug 文件夹，里面包含驱动文件 firstprg_drv.ko、应用测试程序 main.c、源文件夹 srcc.o、目标文件夹 obj 以及驱动源文件夹 driver，如图 22 所示。

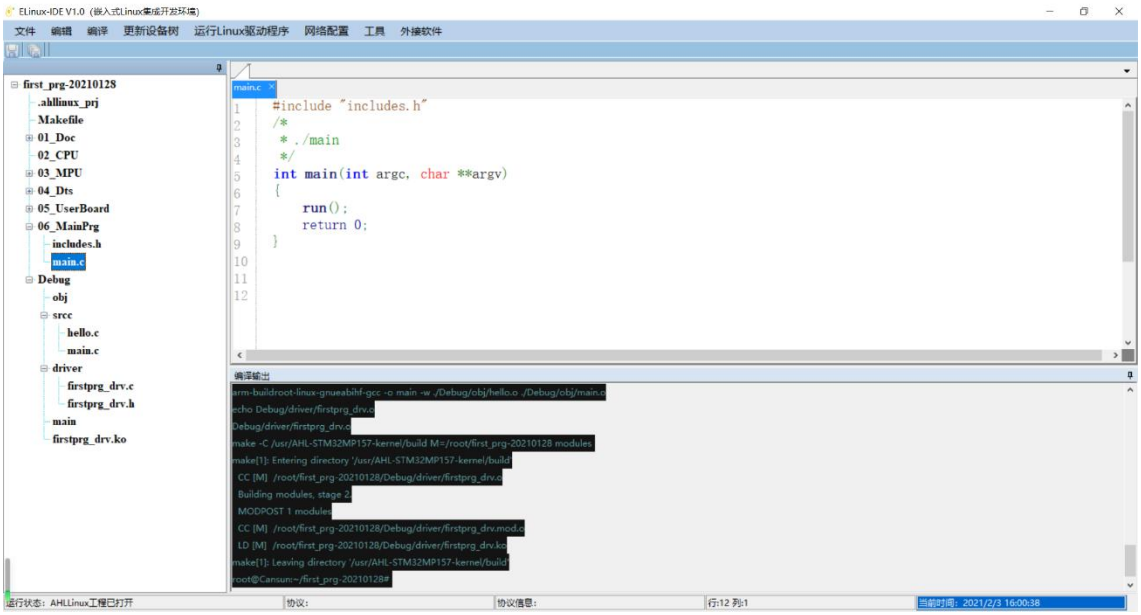

图 22 编译程序

4．运行程序

将一根双母头杜邦线的一头连接发光二极管的较长一端，另一头连接开发板上的 PH6 引脚（LCD 下方

十二个引脚中从上往下从左往右数的第一个引脚）。将另一根双母头杜邦线的一头连接光二极管的较短一端，另一头连接开发板上的 GND 引脚。

点击 ELinux-IDE 上方的“运行 Linux 驱动程序”→“网络运行”按钮，稍等片刻，若看到发光二极管亮起，则说明运行成功。

附录-20210219

附录 1 虚拟机 Ubuntu 编译环境搭建

1．VMWare 上安装 Ubuntu18.04

VMWare 安装可参考教程 http://www.zhanshaoyi.com/15261.html。

在 VMWare 上安装 Ubuntu18.04 可参考教程 https://blog.csdn.net/qq_33287871/article/details/99212352。

安装完 Ubuntu18.04 后，屏幕会比较小，而且不能从主机复制文件到虚拟机内，所以需要安装 Vmware Tools，可参考教程 https://ywnz.com/linuxjc/3144.html。

2．安装 SSH 协议必备服务 openssh、openssl

打开 Ubuntu 终端，输入命令 sudo apt install openssh-server libssl-dev，安装成功后可在终端输入命令 ssh –V 查看，如图 1-1 所示。

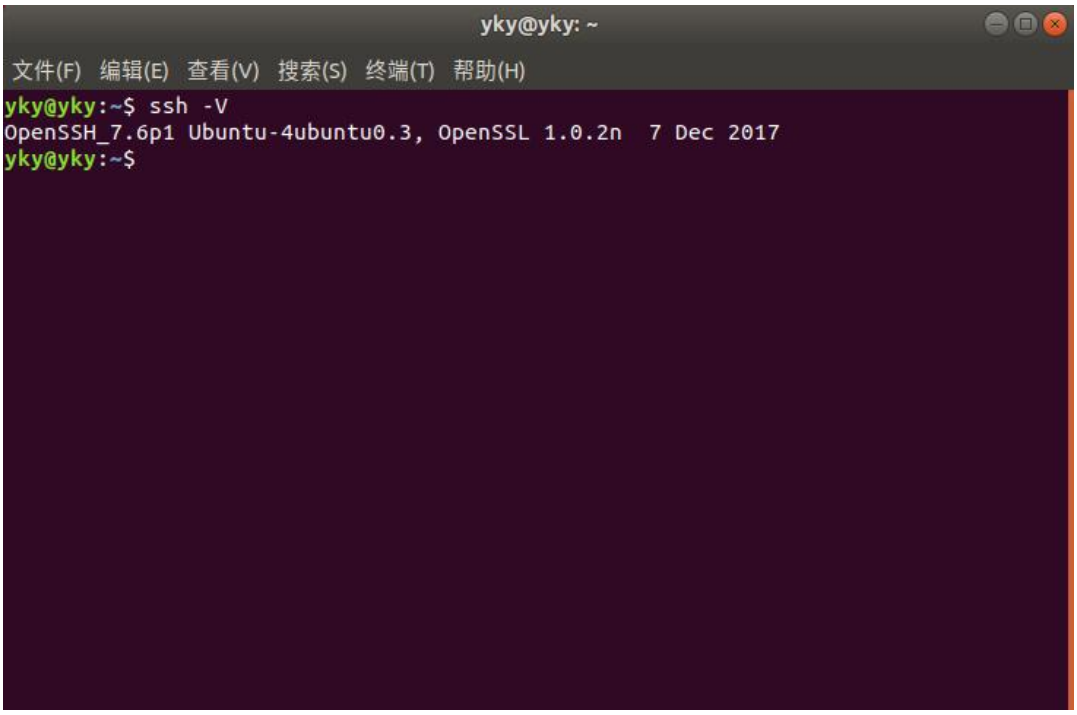

图 1-1 安装 openssh、openssl

3．配置虚拟机桥接模式

此步需要先关闭虚拟机 Ubuntu。

在 VMware Workstation 左上方菜单栏中选择“编辑”→“虚拟网络编辑器”，选中 VMnet0，在 VMnet 信息中将模式修改为“桥接模式”，并在“已桥接至”一栏选择对应的以太网网卡，如我的为以太网 4，故对应的以太网网卡为 Realtek USB GbE Family Controller，如图 1-2 所示。

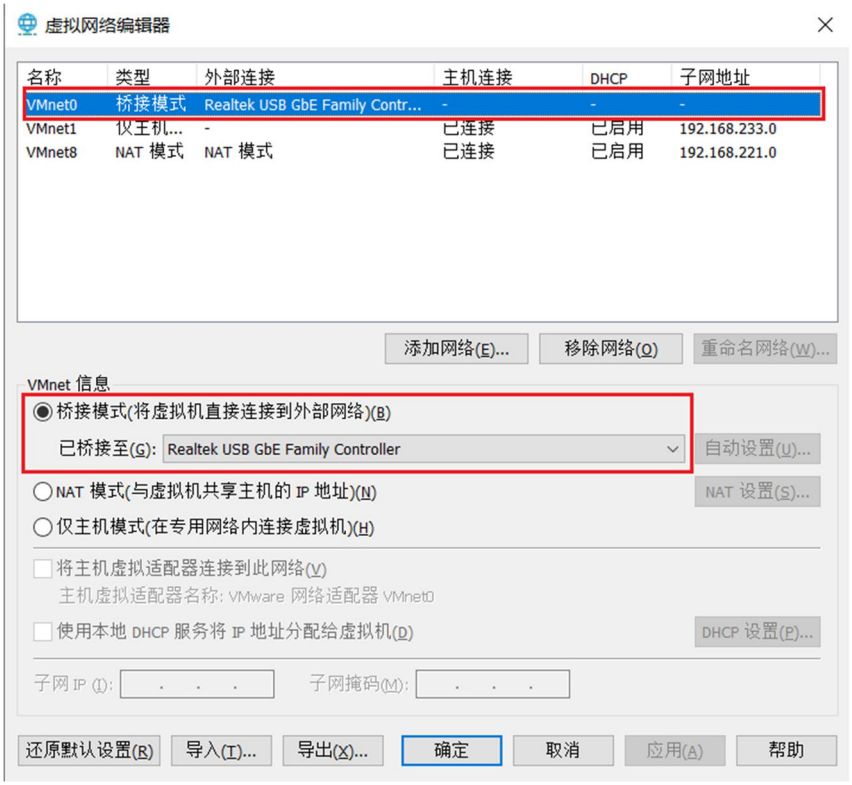

图 1-2 安配置虚拟机桥接模式

重新打开虚拟机 Ubuntu 即可。

3．配置静态 IP

打开虚拟机 Ubuntu 终端，输入命令 sudo gedit /etc/netplan/01-network-manager-all.yaml，在打开的文档中添加如下代码。

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    ens33: #配置的网卡名称,使用 ifconfig -a 查看得到
      dhcp4: no #dhcp4 关闭
      addresses: [192.168.31.80/24] #设置本机 IP 及掩码
      gateway4: 192.168.31.1 #设置网关
      nameservers:
        addresses: [192.168.31.1] #设置 DNS
```

在终端输入命令 sudo netplan apply 应用修改。

打开电脑主机 cmd 终端，输入 ping 192.168.31.80，若能 ping 通，则说明配置静态 IP 成功，成功界面如图 1-3 所示。

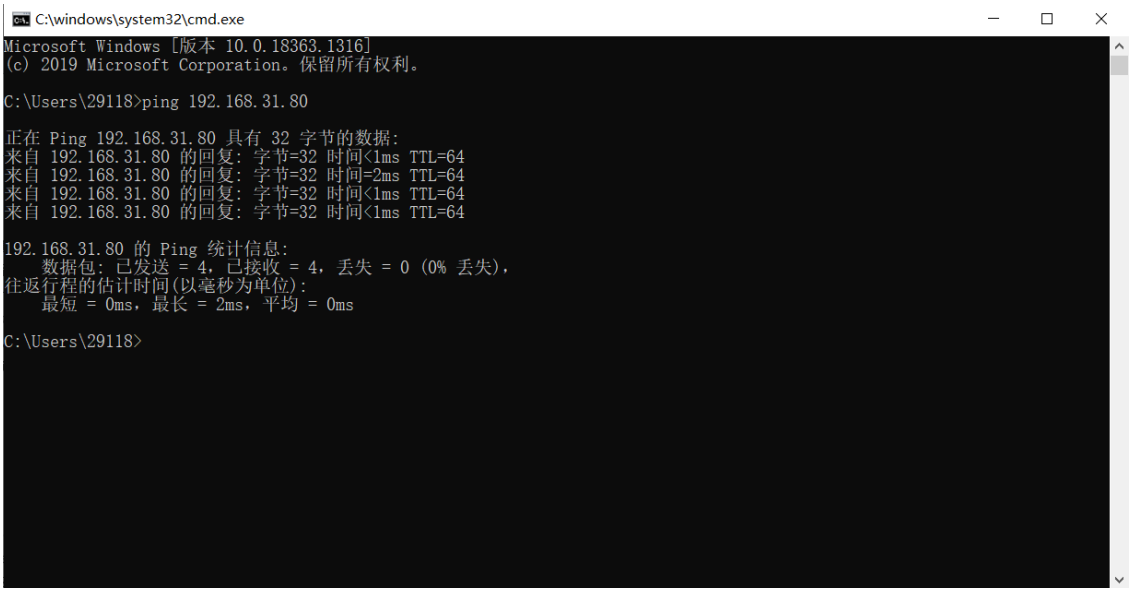

图 1-3 连接测试

4．修改 ssh 配置文件

在终端输入 sudo gedit /etc/ssh/sshd_config，在文件末尾增加代码：

```
Ciphers
3des-cbc,aes128-cbc,aes192-cbc,aes256-cbc,rijndael-cbc@lysator.liu.se,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com

MACs
hmac-sha1,hmac-sha1-96,hmac-sha2-256,hmac-sha2-512,hmac-md5,hmac-md5-96,umac-64@openssh.com,umac-128@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha1-96-etm@openssh.com,hmac-sha2-256-etm@openssh.com,,mac-sha2-512-etm@openssh.com,hmac-md5-etm@openssh.com,hmac-md5-96-etm@openssh.com,umac-64-etm@openssh.com,umac-128-etm@openssh.com

KexAlgorithms
diffie-hellman-group1-sha1,diffie-hellman-group14-sha1,diffie-hellman-group-exchange-sha1,diffie-hellman-group-exchange-sha256,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group1-sha1,curve25519-sha256@libssh.org
```

如图 1-4 所示。

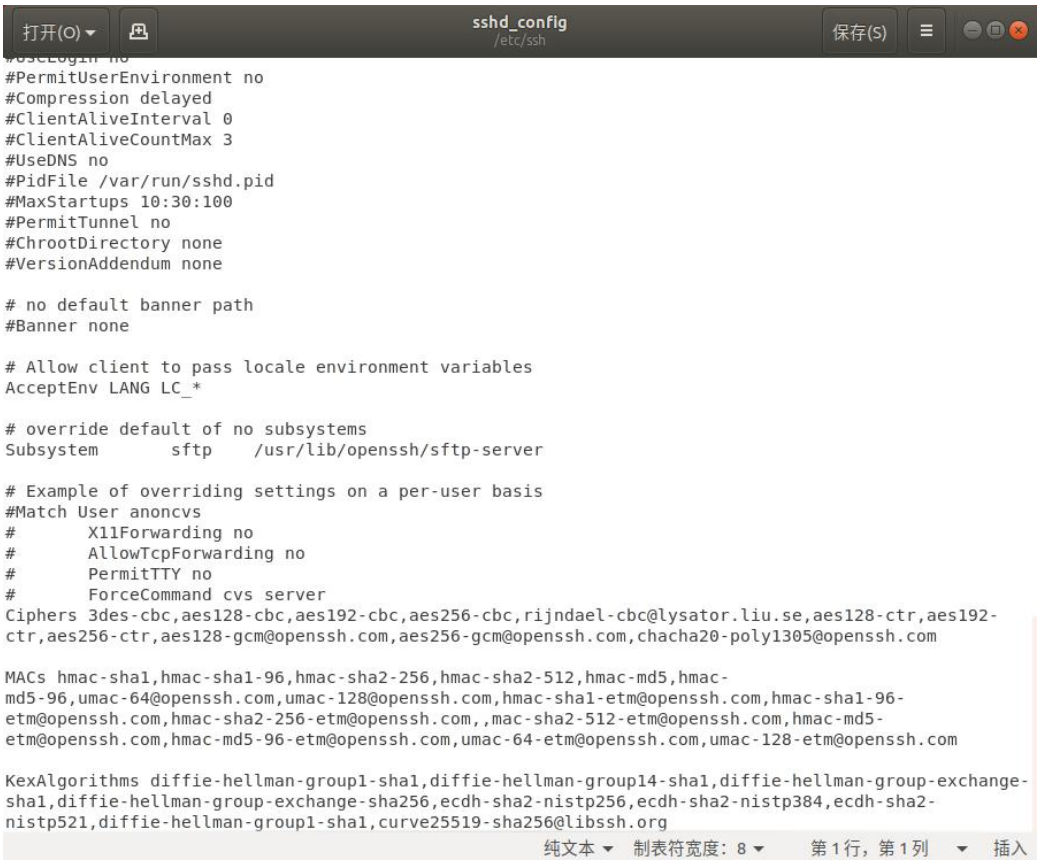

图 1-4 修改 ssh 配置文件

5．配置交叉编译工具链

将 05_Source/arm-buildroot-linux-gnueabihf_sdk-buildroot.tar.xz 文件复制（需要已经装好 Vmware Tools）到 Ubuntu 的/home/（自己的用户名）目录下（最好是在这里，放置在其他地方也可以），输入命令“tar –xvf arm-buildroot-linux-gnueabihf_sdk-buildroot.tar.xz”进行解压。

打开终端，输入命令 sudo gedit ~/.bashrc，在文件末尾增加代码：

```
export ARCH=arm
export CROSS_COMPILE=arm-buildroot-linux-gnueabihf-
export PATH=$PATH:/home/（自己的用户名）/arm-buildroot-linux-gnueabihf_sdk-buildroot/bin
```

保存，然后在终端输入命令 source ~/.bashrc 保存配置，如图 1-5 所示。

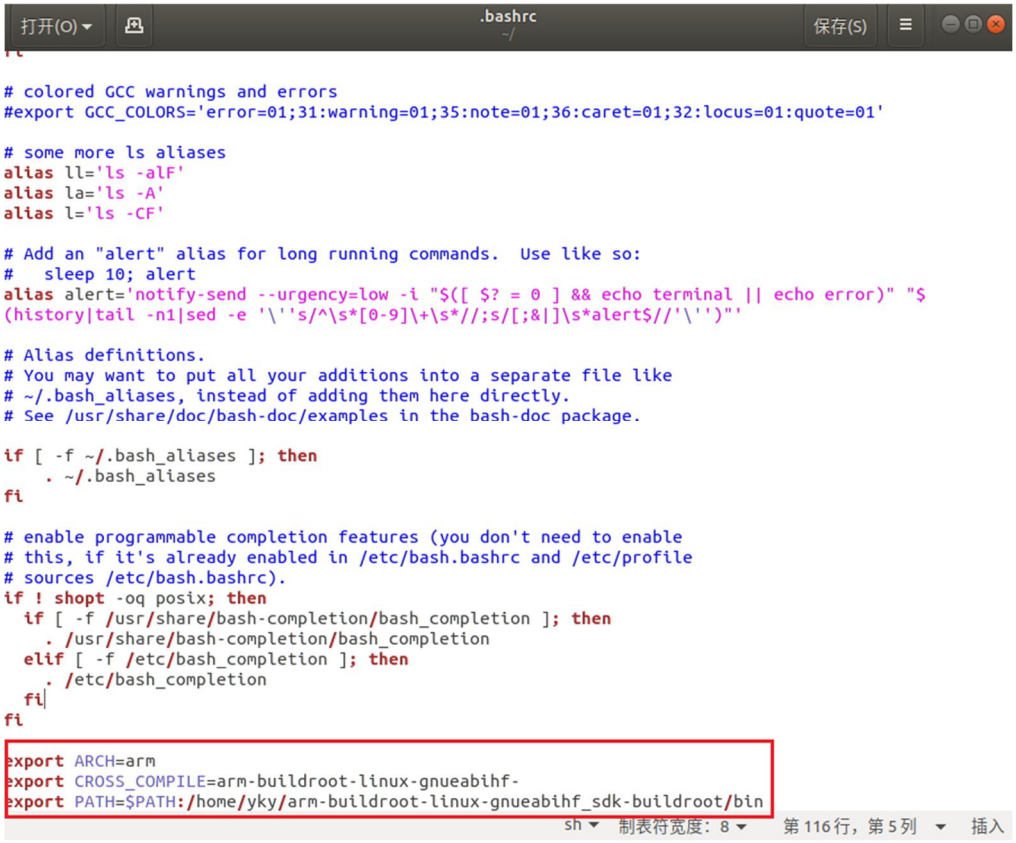

图 1-5 配置交叉编译工具链

6．解压 Linux 内核

将 05_Sources/AHL-STM32MP157-kernel.zip 文件复制（需要已经装好 Vmware Tools）到虚拟机 Ubuntu 的/home/（你的用户名）目录下，然后打开终端逐行输入和运行如下几行命令：

```
sudo su
cp /home/（你的用户名）/AHL- STM32MP157-kernel.zip /usr
cd /usr
unzip AHL- STM32MP157-kernel.zip –d AHL- STM32MP157-kernel
```

至此虚拟机 Ubuntu 开发环境配置完成。

附录 2 ELinux-IDE 使用指南

本附录为 ELinux-IDE 软件的使用指南，包含了该软件各个部分功能的示例演示。该软件的主界面如图 2-1 所示。

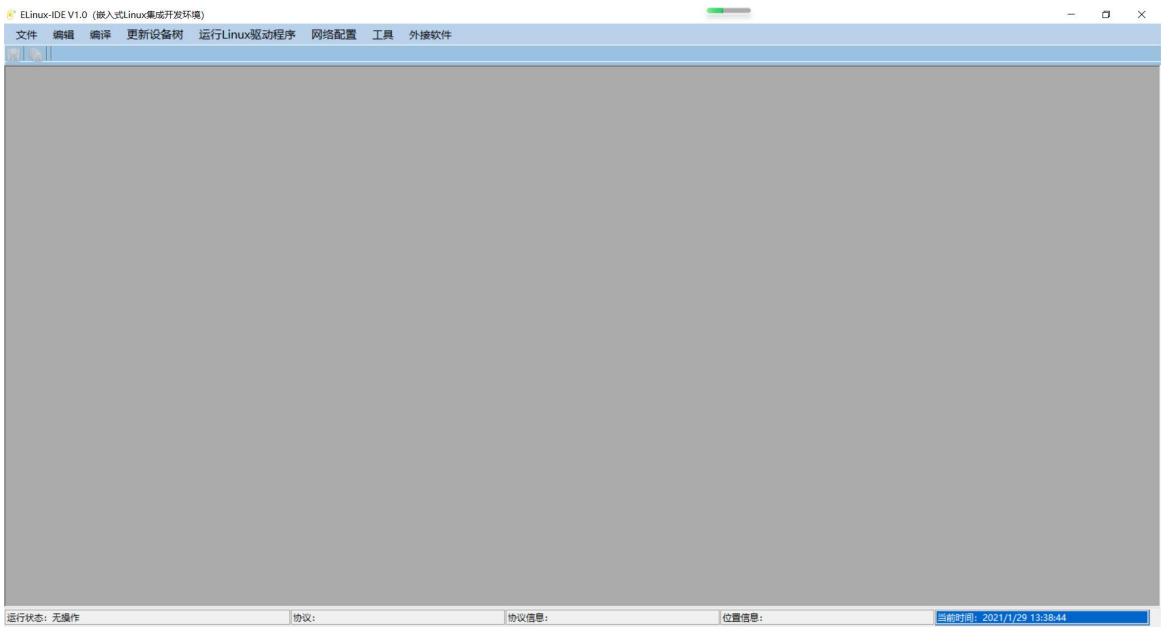

图 2-1 ELinux 主界面

1．导入工程

运行 ELinux-IDE，单击“文件”→“导入工程”，选择工程 02_Soft\first_prg-20210128。导入工程后，左侧为工程树形目录，右边为文件内容编辑区，初始显示 main.c 文件的内容，如图 2-2 所示。

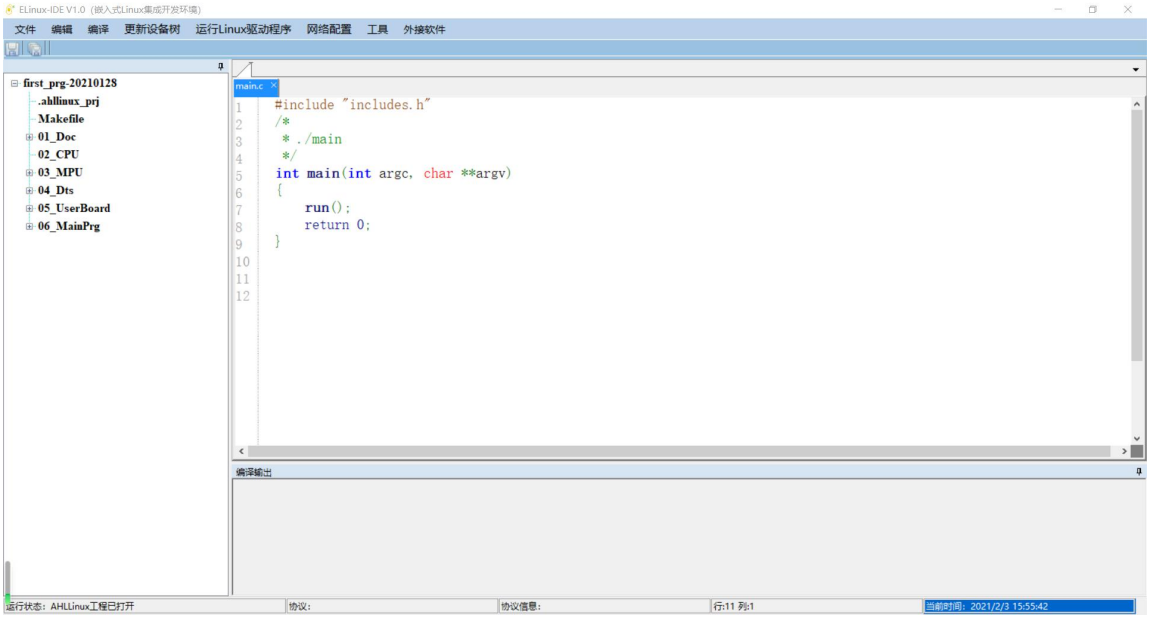

图 2-2 工程在 ELinux-IDE 中的示意图

2．配置网络

由于 ELinux-IDE 的编译方式有虚拟机 Linux 与本地 Linux 编译两种方式，其中使用虚拟机 Linux 编译时需要获得虚拟机的 IP 地址、用户名和密码；而 ELinux-IDE 运行 Linux 驱动程序有串口运行与网络运行两种方式，其中网络运行需要获得开发板的 IP 地址、用户名和密码。

其中虚拟机的 IP 地址可通过打开虚拟机终端，输入命令 ifconfig 查看，如图 2-3 所示，红框中即为虚拟

机的 IP 地址。

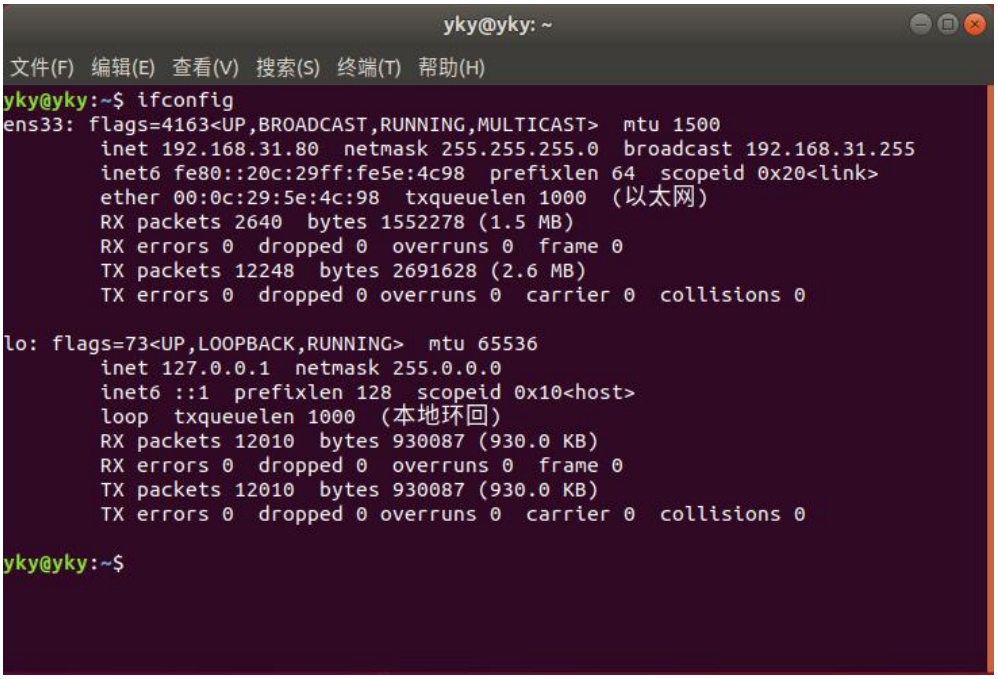

图 2-3 获取虚拟机 IP

开发板的 IP 地址可通过串口发送命令获取。在网络连通的情况下，打开串口工具 XCOM，在发送框输入命令“ifconfig”，可获取开发板 IP，如图 2-4 所示。

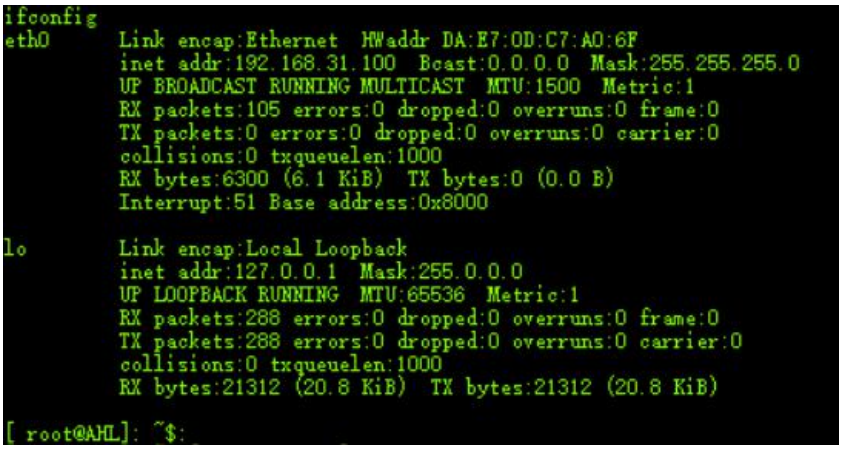

图 2-4 获取开发板 IP

单击“网络配置”进行网络配置，在弹出的对话框中填入对应虚拟机的 IP、用户名、密码以及开发板的 IP、用户名、密码，如图 2-5 所示。

图 2-5 ELinux-IDE 网络配置

3．编译工程

在打开工程、显示文件内容并配置好网络的前提下，可编译工程，编译后会在工程目录下生成 Debug 文件夹，里面包含以 ko 为后缀的驱动文件以及名为 main 的应用测试文件。

编译工程有如下两种方式。

1）虚拟机编译

在开启虚拟机的前提下，单击“编译”→“虚拟机编译 Linux 驱动”，开始编译，编译完成如图 2-6 所示。

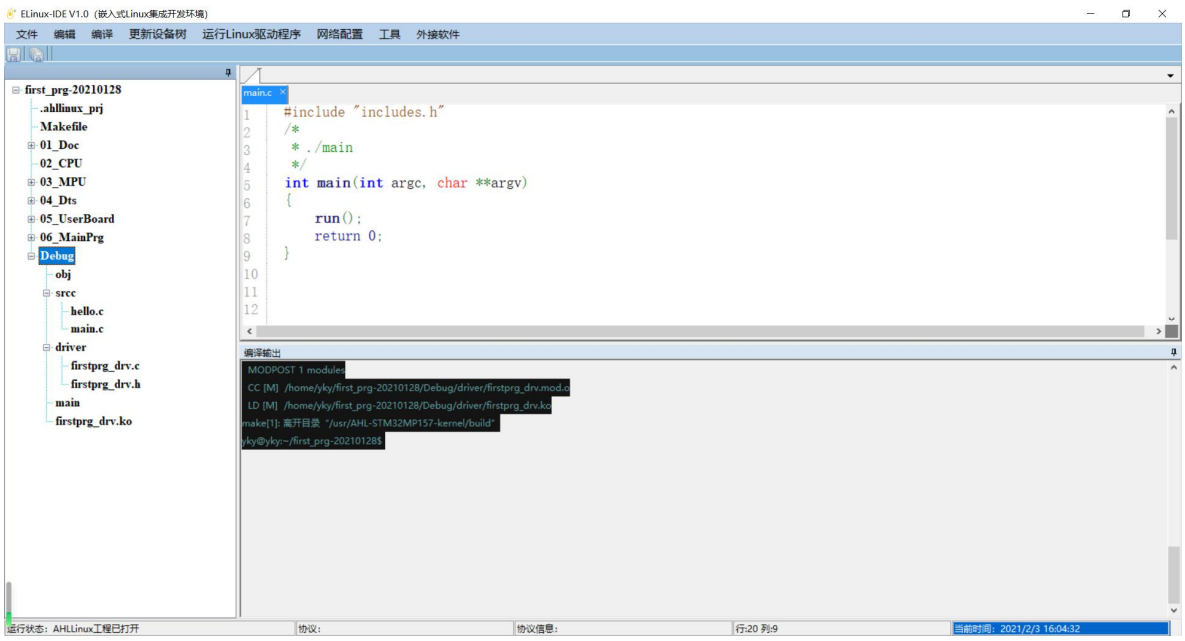

图 2-6 虚拟机编译

2）本地 Linux 编译

在本地 Linux 终端开启且输入命令 sudo su 进入超级用户模式的前提下，单击“编译”→“本地编译 Linux 驱动”，开始编译，编译完成如图 2-7 所示。

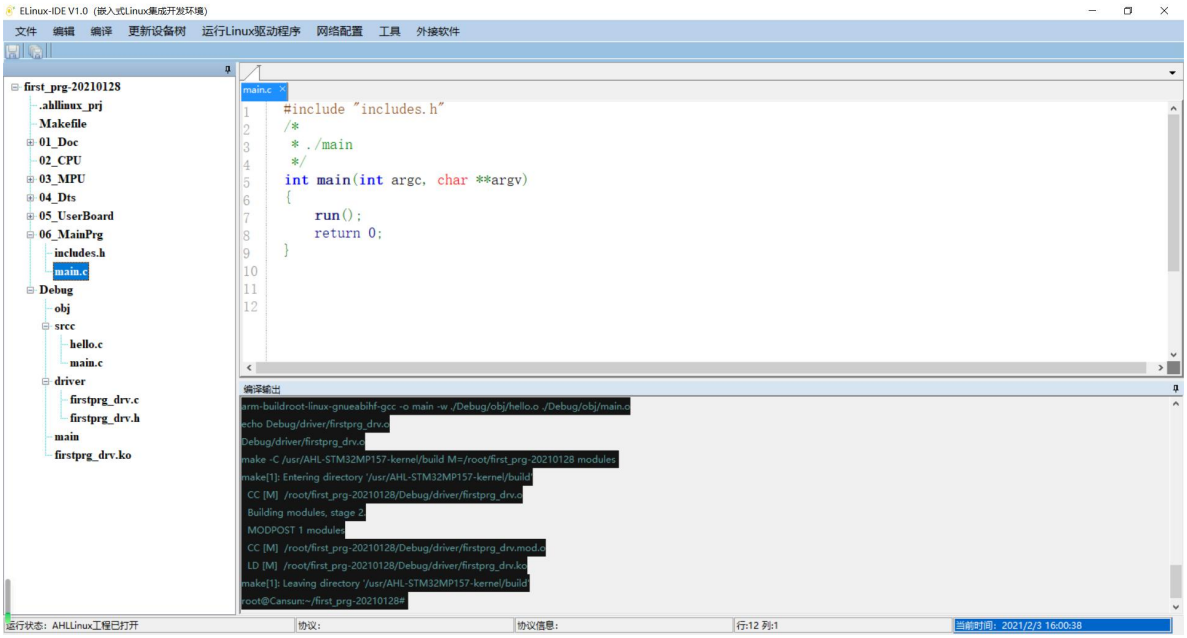

图 2-7 本地 Linux 编译

4．运行工程

编译完成后，即可运行工程。此处的示例工程功能为点亮一个发光二极管，需要用一根双母头杜邦线连接二极管长端和开发板 PH6 引脚，用另一根双母头杜邦线连接二极管短端和开发板 GND 引脚。

1）串口运行

单击“运行 Linux 驱动程序”→“串口运行”，在弹出的窗体中选择对应的串口，如此处为 COM29，点击“打开串口”，点击“发送并运行”，会显示文件上传进度，如图 2-8 所示。等待文件上传完成后会自动运行，若发光二极管闪烁，说明运行成功。

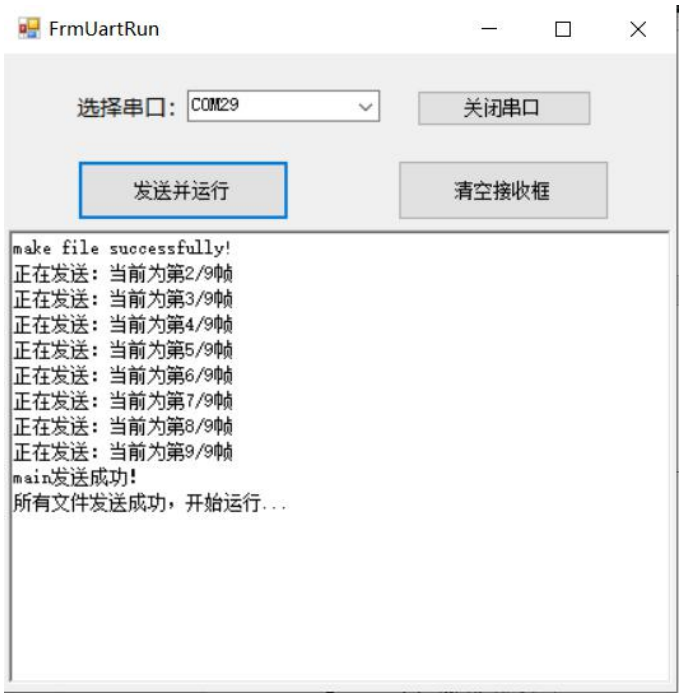

图 2-8 串口运行

2）网络运行

在网络配置及接线成功的前提下，单击“运行 Linux 驱动程序”→“网络运行”，若发光二极管闪烁，说明运行成功。

5．更新设备树

在打开工程、显示文件内容并配置好网络的前提下，可进行开发板的设备树更新，开发板的设备树文件存放在工程的 03_Dtb 文件夹下，以 dts 为后缀的文件为设备树源文件，以 dtb 为后缀的文件为更新后生成的设备树二进制文件。此处以将小灯的 GPIO 引脚从 PH6 改为 PF11 为例，进行设备树更新。

打开 03_Dtb 下的 ahl_stm32mp157.dts 文件，将节点 firstprg 中的 led-gpios 值改为“<&gpiof 11 GPIO_ACTIVE_HIGH>”，将节点&pinctrl 中的 firstprg_pin 中的 pin 中的 pinmux 的值改为“<STM32_PINMUX('F', 11, GPIO)>”，保存。

更新设备树有如下两种方式。

1）虚拟机更新

在虚拟机已开启的前提下，单击“更新设备树”→“虚拟机更新”，开始虚拟机更新设备树，等待直至提示更新设备树完成，如图 2-9 所示，重新启动开发板后即可更新成功。

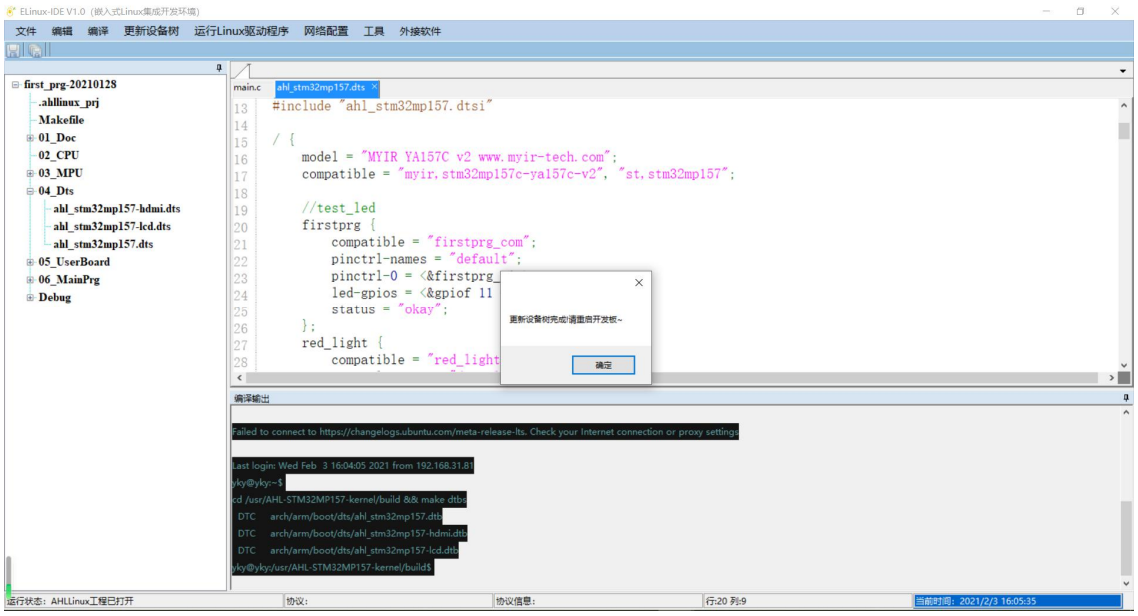

图 2-9 虚拟机更新设备树

此时将外接二极管长端的杜邦线从接 PH6 引脚变为接 PF11 引脚，再次运行工程，可发现二极管闪烁。

2）本地 Linux 更新

在本地 Linux 终端开启且输入命令 sudo su 进入超级用户模式的前提下，单击“更新设备树”→“本地 Linux 更新”，开始本地 Linux 更新设备树，等待直至提示更新设备树完成，如图 2-10 所示，重新启动开发板后即可更新成功。

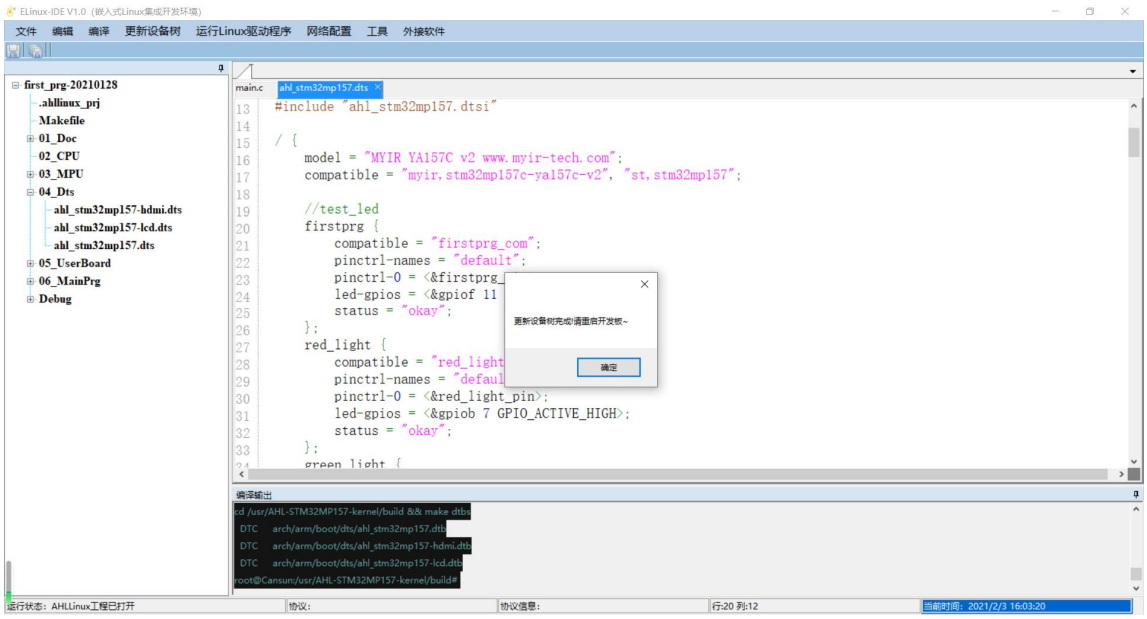

图 2-10 本地 Linux 更新设备树

此时将外接二极管长端的杜邦线从接 PH6 引脚变为接 PF11 引脚（sd 卡接口下方第四排引脚中右边的引脚），再次运行工程，可发现二极管闪烁。

附录 3 Linux 入门

3.1 简介

1．概述

Linux 内核最初只是由芬兰人林纳斯·托瓦兹（Linus Torvalds）在赫尔辛基大学上学时出于个人爱好而编写的。Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。Linux 能运行主要的 UNIX 工具软件、应用程序和网络协议。它支持 32 位和 64 位硬件。Linux 继承了 Unix 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

目前市面上较知名的发行版有：Ubuntu、RedHat、CentOS、Debian、Fedora、SuSE、OpenSUSE、Arch Linux、SolusOS 等。

关于 Linux 的入门教程可参考网址 https://www.runoob.com/linux/linux-tutorial.html。

2．Linux 与 Windows 的比较

目前国内 Linux 更多的是应用于服务器上，而桌面操作系统更多使用的是 Windows。主要区别如表 3-1 所示：

表 3-1 Linux 与 Windows 的比较

比较	Windows	Linux
界面	界面统一，外壳程序固定所有 Windows 程序菜单几乎一致，快捷键也几乎相同	图形界面风格依发布版不同而不同，可能互不兼容。GNU/Linux 的终端机是从 UNIX 传承下来，基本命令和操作方法也几乎一致。
驱动程序	驱动程序丰富，版本更新频繁。默认安装程序里面一般包含有该版本发布时流行的硬件驱动程序，之后所出的新硬件驱动依赖于硬件厂商提供。对于一些老硬件，如果没有了原配的驱动有时很难支持。另外，有时硬件厂商未提供所需版本的 Windows 下的驱动，也会比较头痛。	由志愿者开发，由 Linux 核心开发小组发布，很多硬件厂商基于版权考虑并未提供驱动程序，尽管多数无需手动安装，但是涉及安装则相对复杂，使得新用户面对驱动程序问题（是否存在和安装方法）会一筹莫展。但是在开源开发模式下，许多老硬件尽管在Windows下很难支持的也容易找到驱动。HP、Intel、AMD 等硬件厂商逐步不同程度支持开源驱动，问题正在得到缓解。
使用	使用比较简单，容易入门。图形化界面对没有计算机背景知识的用户使用十分有利。	图形界面使用简单，容易入门。文字界面，需要学习才能掌握。
学习	系统构造复杂、变化频繁，且知识、技能淘汰快，深入学习困难。	系统构造简单、稳定，且知识、技能传承性好，深入学习相对容易。
软件	每一种特定功能可能都需要商业软件的支持，需要购买相应的授权。	大部分软件都可以自由获取，同样功能的软件选择较少。

3．ubuntu 文件系统结构

由于本文档使用的 Linux 操作系统为 Ubuntu，故对 Ubuntu 的文件系统结构进行简要的介绍。

/是一切目录的起点，如大树的主干。其它的所有目录都是基于树干的枝条或者枝叶。在 ubuntu 中硬件设备如光驱、软驱、usb 设备都将挂载到这颗繁茂的枝干之下，作为文件来管理，Ubuntu 根文件系统目录如

图 3-1 所示。

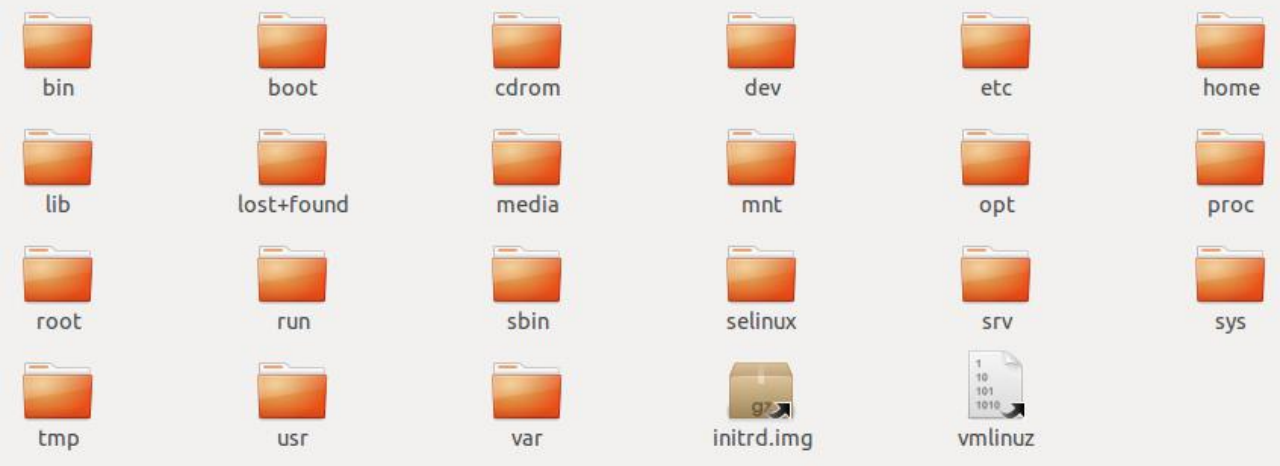

图 3-1 Ubuntu 根文件系统目录

（1）/bin: bin 是 Binary 的缩写。存放系统中最常用的可执行文件（二进制）。

（2）/boot: 这里存放的是 linux 内核和系统启动文件，包括 Grub、lilo 启动器程序。

（3）/dev: dev 是 Device(设备)的缩写。该目录存放的是 Linux 的外部设备，如硬盘、分区、键盘、鼠标、usb 等。

（4）/etc: 这个目录用来存放所有的系统管理所需要的配置文件和子目录，如 passwd、hostname 等。

（5）/home: 用户的主目录，在 Linux 中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。

（6）/lib: 存放共享的库文件，包含许多被/bin 和/sbin 中程序使用的库文件。

（7）/lost+found: 这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些零散文件。

（8）/media: ubuntu 系统自动挂载的光驱、usb 设备，存放临时读入的文件。

（9）/mnt: 作为被挂载的文件系统得挂载点。

（10）/opt: 作为可选文件和程序的存放目录，主要被第三方开发者用来简易安装和卸载他们的软件。

（11）/proc: 这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。这里存放所有标志为文件的进程，比较 cpuinfo 存放 cpu 当前工作状态的数据。

（12）/root: 该目录为系统管理员，也称作超级权限者的用户主目录。

（13）/sbin: s 就是 Super User 的意思，这里存放的是系统管理员使用的系统管理程序，如系统管理、目录查询等关键命令文件。

（14）/ srv: 存放系统所提供的服务数据。

（15）/sys: 系统设备和文件层次结构，并向用户程序提供详细的内核数据信息。

（16）/tmp: 这个目录是用来存放一些临时文件的，所有用户对此目录都有读写权限。

（17）/usr: 存放与系统用户有关的文件和目录。

3.2 Linux 命令大全

Linux 具有丰富的 Shell 命令，包括文件管理、文档编辑、文件传输、磁盘管理、磁盘维护、网络通信、系统管理、系统设置、备份压缩以及设备管理等。

具体的命令以及使用规则可参考 https://www.runoob.com/linux/linux-command-manual.html。

3.3 嵌入式 Linux 系统入门

1．简介

嵌入式 Linux 是将日益流行的 Linux 操作系统进行裁剪修改，使之能在嵌入式计算机系统上运行的一种操作系统。嵌入式 linux 系统多用于定制开发专用设备，功能固定，对于不同的行业，根据需求对软硬件进行裁剪选配，这也是嵌入式 Linux 系统广泛应用的特点之一。

嵌入式 Linux 系统具有如下几个特点：

（1）C 语言即可入门。嵌入式 Linux 开发，大多使用 C 或 C++，底层内核代码以及驱动代码大多由 C 语言实现，内核提供的系统 API 均使用的是 C 语言接口，因此会 C 语言编程即可入门。

（2）命令行。命令行在 Linux 开发中，是与系统沟通的最主要方式，其常用命令与 PC 端和设备端基本一致，这些命令均由 C 语言实现。在嵌入式 Linux 系统中，这些命令均来自于 busybox 工具集。

（3）网络。Linux 具有强大的网络功能，有完整的网络通信功能，易加入 WIFI、4G 等通信方式。

（4）远程运维。基于网络，嵌入式 Linux 系统很容易实现远程运维，可通过内网穿透技术、或基于 SSH2 协议实现远程跨公网访问设备，便于开发人员进行测试、调试、升级、问题查找等。嵌入式 Linux 系统还具有丰富的远程传输文件方式，如 ftp、scp、wget、xmodule 等方式。

（5）成本。随着技术发展，芯片的价格也在不断下降，嵌入式 linux 系统硬件成本也在不断下降，使得其应用的更加广泛。

2．嵌入式 Linux 系统架构

嵌入式 Linux 的系统架构如图 3-2 所示。

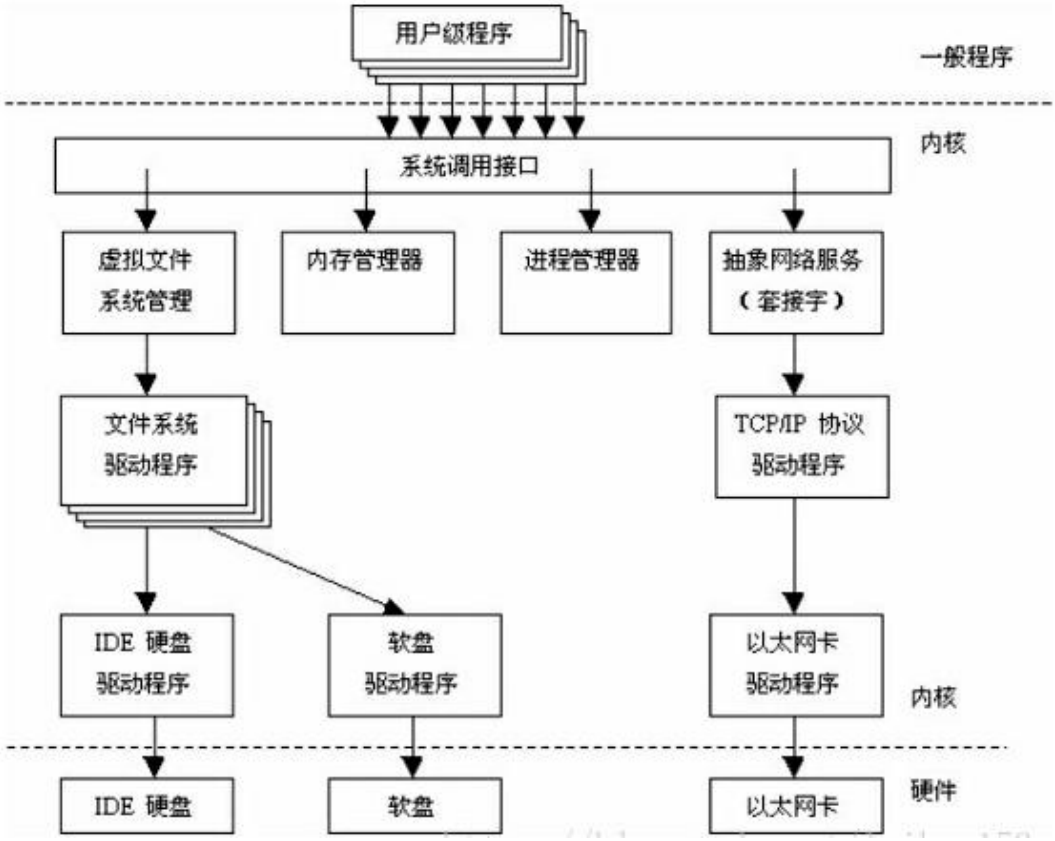

图 3-2 嵌入式 Linux 系统架构

嵌入式 Liunx 系统分为三层：用户层、内核层、硬件层

（1）应用层是一些应用程序和库，是面向用户的，如命令、QQ 等应用程序

（2）内核层的主要功能是设备驱动、进程管理、内存管理、文件系统和网络通信。

内核与应用程序之间是系统调用接口函数，内核与硬件之间的接口是驱动程序，驱动程序负责硬件操作，内核提供了驱动程序的添加机制，便于开发人员将驱动代码添加到内核中。

（3）硬件层是以信号为对象，完成各种信号之间的相互转换。

嵌入式系统硬件电路是由微处理器、存储器、输入输出设备、通信与扩展接口构成。

3．标准 Linux 启动过程

嵌入式 Linux 的标准启动过程分为五步，ROM Code→TF-A（FSBL）→U-Boot（SSBL）→Linux kernel →文件系统→Linux user。各步骤实现的具体操作如下：

1）ROM Code（在 ROM（<128KB）中）

基本时钟树初始化、从启动设备（存储器或串口）加载第一级启动加载程序 FSBL。

2）FSBL（First Stage Boot Loader）（在内置 RAM（<256KB）中）

完成时钟树初始化，初始化外部 RAM（DDR，LpDDR）控制器，从启动设备（存储器或串口）加载第二级启动加载程序 SSBL。

3）SSBL（Second Stage Boot Loader）（在外置 RAM（>=512MB）中）

从存储器或以太网加载启动文件系统（bootfs），显示开机画面，加载 Linux kernel（uImage）以及设备树（device tree blob,*.dtb）。

4）Linux kernel（在外置 RAM（>=512MB）中）

Linux 内核初始化，挂载根文件系统，加载用户空间初始化进程。

5）Linux user space（在外置 RAM（>=512MB）中）

加载用户空间服务和应用。