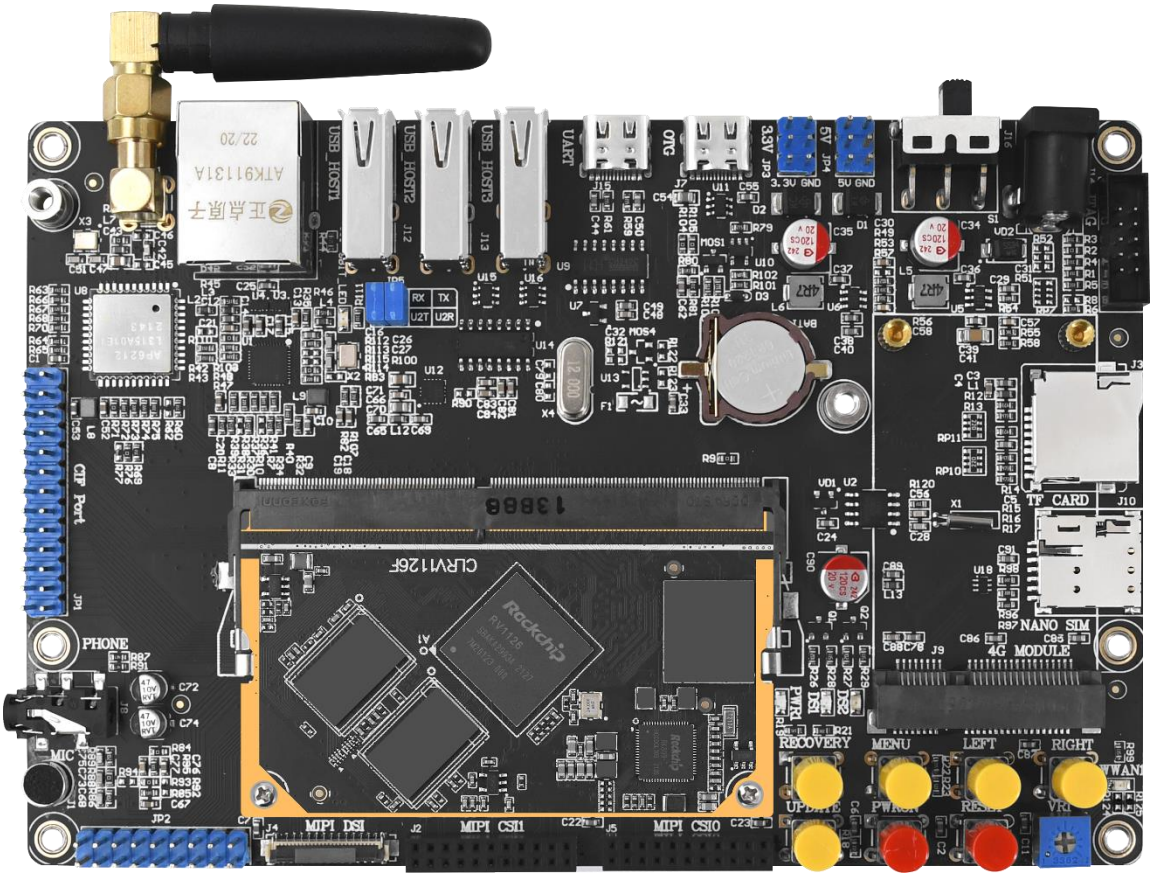
# ATK-DLRV1126 系统开发手册 V1.5



正点原子

正点原子公司名称 ：广州市星翼电子科技有限公司
原子哥在线教学平台：www.yuanzige.com
开源电子网 / 论坛 ：http://www.openedv.com/forum.php
正点原子淘宝店铺 ：https://openedv.taobao.com
正点原子官方网站 ：www.alientek.com
正点原子 B 站视频 ：https://space.bilibili.com/394620890

电话：020-38271790 传真：020-36773971

请关注正点原子公众号，资料发布更新我们会通知。

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

扫码关注正点原子公众号

扫码下载“原子哥”APP

# 文档更新说明

| 版本 | 版本更新说明 | 负责人 | 校审 | 发布日期 |
|---|---|---|---|---|
| V1.0 | **初稿：** | 正点原子 Linux 团队 | 正点原子 Linux 团队 | 2022.09.6 |
| V1.1 | **初稿：重新修改第 4 章节** | 正点原子 Linux 团队 | 正点原子 Linux 团队 | 2022.11.17 |
| V1.2 | **添加了第五章节 RKMedia 编译和使用**<br>**修改名字为【正点原子】ATK-DLRV1126 系统开发手册 V1.x**<br>**添加了 3.10 小节** | 正点原子 Linux 团队 | 正点原子 Linux 团队 | 2022.12.07 |
| V1.3 | **添加第六章节和附录 A** | 正点原子 Linux 团队 | 正点原子 Linux 团队 | 2023.01.14 |
| V1.4 | **添加 dl 包解压的注意事项** | 正点原子 Linux 团队 | 正点原子 Linux 团队 | 2023.05.19 |
| V1.5 | **1. 修改 A1.2 小节**<br>**2. 添加 4.6.5 小节，A2 小节** | 正点原子 Linux 团队 | 正点原子 Linux 团队 | 2023.08.17 |

# 前言

## 免责声明

本文档所提及的产品规格和使用说明仅供参考，如有内容更新，恕不另行通知；除非有特殊约定，本文档仅作为产品指导，所作陈述均不构成任何形式的担保。

为了得到最新版本的产品信息，请用户定时访问正点原子资料下载中心或者与淘宝正点原子旗舰店客服联系索取。感谢您的包容与支持。

本文档不适合初学者，本文档不适合初学者，本文档不适合初学者。

# 目录

# 第一篇 环境搭建篇

本篇主要讲解开发环境搭建，如何安装虚拟机，在虚拟机上安装 Ubuntu 操作系统。

# 第一章 VMware 虚拟机安装

Linux 的开发需要在 Linux 系统下进行，这就要求我们的 PC 主机安装 Linux 系统，本书我们选择 Ubuntu 这个 Linux 发行版系统。本章讲解如何安装虚拟机，以及如何在虚拟机中安装 Ubuntu 系统，安装完成以后如何做简单的设置。如果已经对于虚拟机以及 Ubuntu 基础操作已经熟悉的话就可以跳过本章。

## 1.1 安装虚拟机软件 VMware

不是安装 Ubuntu 吗？怎么要先安装虚拟机呢？虚拟机是个啥？相信大部分第一次安装 Ubuntu 的人都会有这个疑问。我不能直接安装 Ubuntu 吗？能不能不要虚拟机呢？答案是肯定可以的！直接在电脑上安装 Ubuntu 以后你的电脑就是一个真真正正的 Ubuntu 电脑了，你可以再安装一个 Windows 系统，这样你的电脑就是双系统了，在开机的时候可以选择不同的系统启动。但是这样的话会有一个问题，那就是你每次只能选择其中的一个系统启动，要么 Windows 要么 Ubuntu，但是我们再开发的时候很多时候需要在 Windows 和 Ubuntu 下来回切换，Windows 系统下的软件资源要比 Ubuntu 下丰富的多，这个就涉及到两个系统切换问题，显然如果你直接在电脑上安装 Ubuntu 以后就没法做到，因为你每次开机只能在 Windows 和 Ubuntu 下二选一。如果 Ubuntu 系统能作为 Windows 下的一个软件就好了，我们默认启动 Windows 系统，需要用到 Ubuntu 的话直接打开这个软件就行了。这个当然是可以的！这里就要借助虚拟机了，虚拟机顾名思义就是虚拟出一个机器，然后你就可以在这个机器上安装任何你想要的系统，相当于再克隆出一个你的电脑，这样在主机上运行 Windows 系统，当我们需要用到 Ubuntu 的话就打开安装有 Ubuntu 系统的虚拟机。

虚拟机的实现我们可以借助其他软件，比如 Vmware Workstation，Vmware Workstation 是收费软件，免费的虚拟机软件有 Virtualbox。本书我们使用 Vmware Workstation 软件来做虚拟机。Vmware Workstation 软件可以在 Wmware 官网下载，下载地址：https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html，当前最新的版本是 Vmware Workstation Pro 16，我们下载 Windows 版本的，如图 1.1.1 所示：

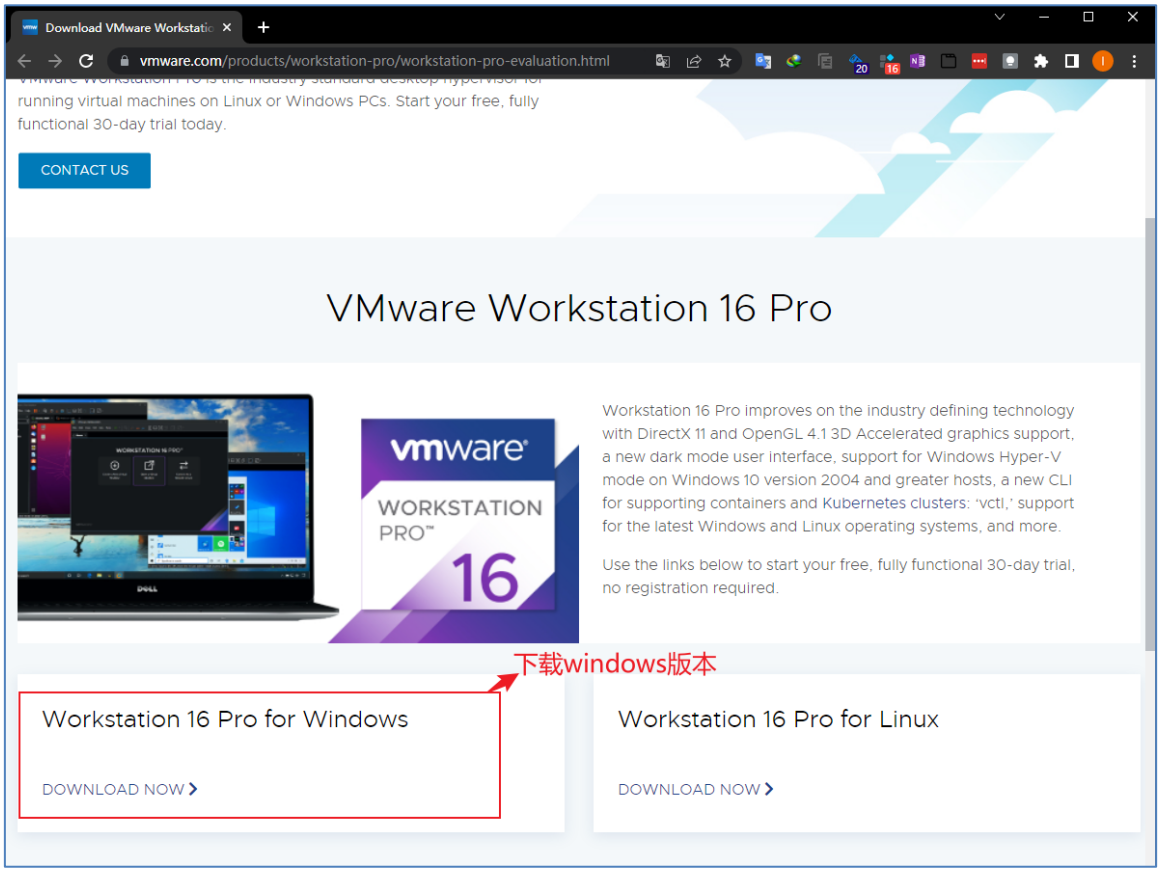



图 1.1.1 Vmware 下载页

我们已经在开发板光盘里面提供了 Vmware Workstation 软件，大家可以直接使用，在光盘

目录：开发板光盘 A-基本资料→4、软件→VMware-workstation-full-16.2.3-19376536.exe。WMware Workstation 的安装和普通软件安装一样，双击 VMware-workstation-full-16.2.3-19376536.exe 进入安装界面，如图 1.1.2 所示：

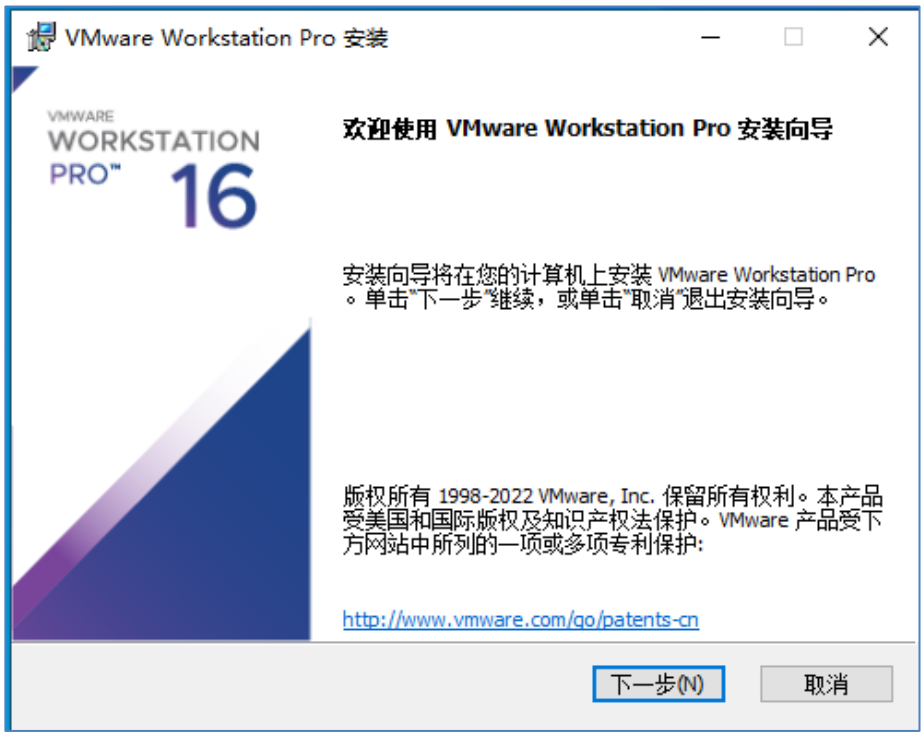



图 1.1.1 VMware 安装界面

点击图 1.1.2 中的“下一步”，进入图 1.1.3 所示步骤：

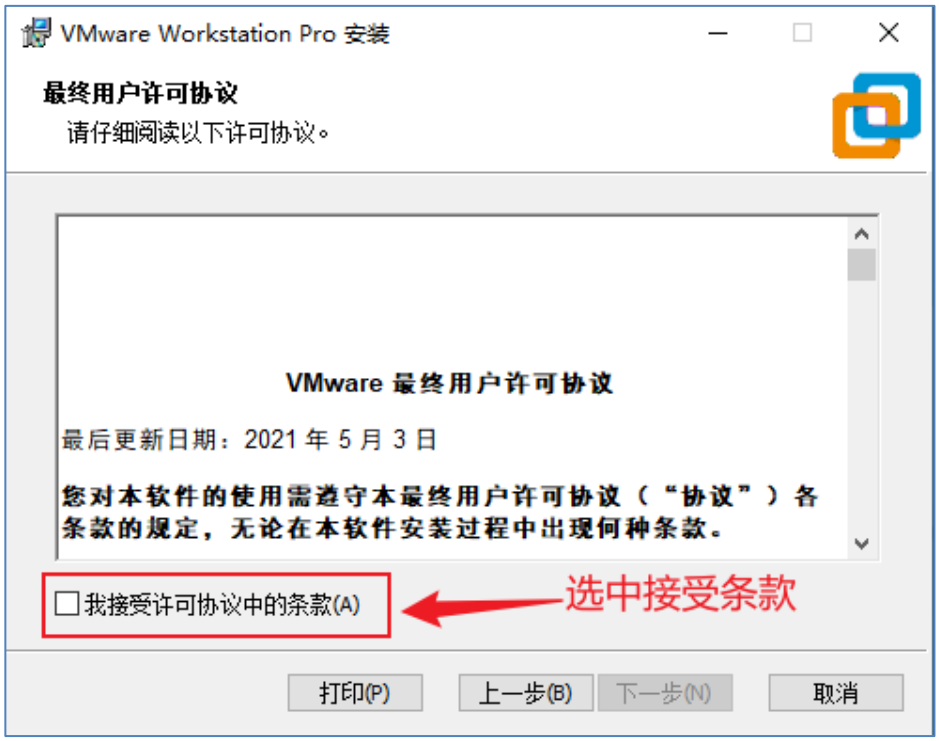



图 1.1.3 VMware 条款

先选择图 1.1.3 中的“我接受许可协议中的条款”，然后在选择“下一步”，进入图 1.1.4 所

示步骤：

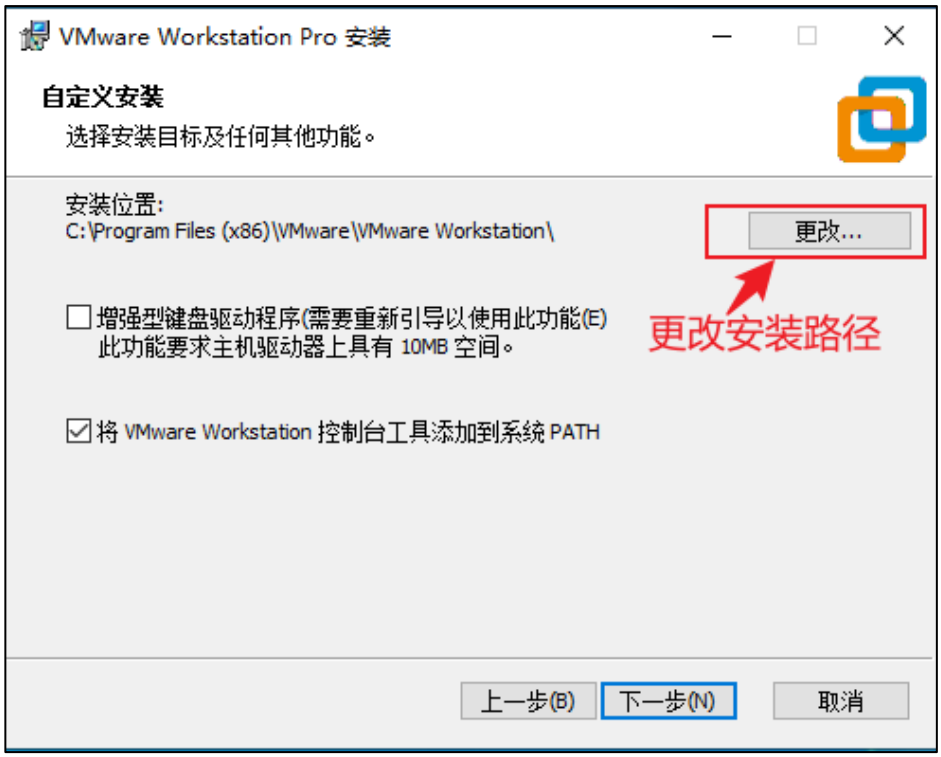



图 1.1.4 选择安装路径

图 1.1.4 中选择软件的安装路径，点击“更改”按钮，然后根据自己的实际需要选择合适路径即可,我的安装路径如图 1.1.5 所示：

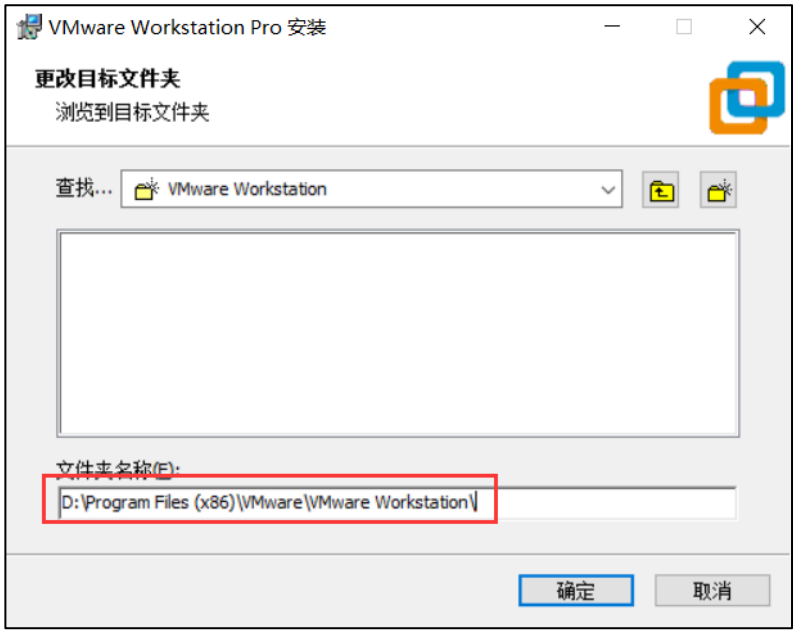



图 1.1.5 安装路径

选择好路径以后点击图 1.1.5 中的“确定”按钮，然后回到图 1.1.4 所示界面，点击图 1.1.4 中的“下一步”，进入图 1.1.6 所示界面：

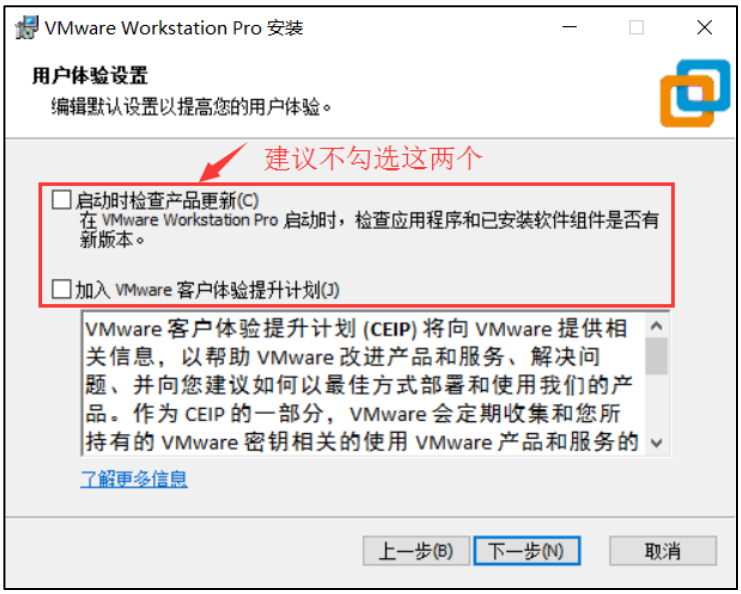



图 1.1.6 检查更新界面

在图 1.1.6 中，会有两个复选框，默认都是选中的，建议不要选中！然后点击图 1.1.6 中的“下一步”按钮，进入图 1.1.7 所示界面：

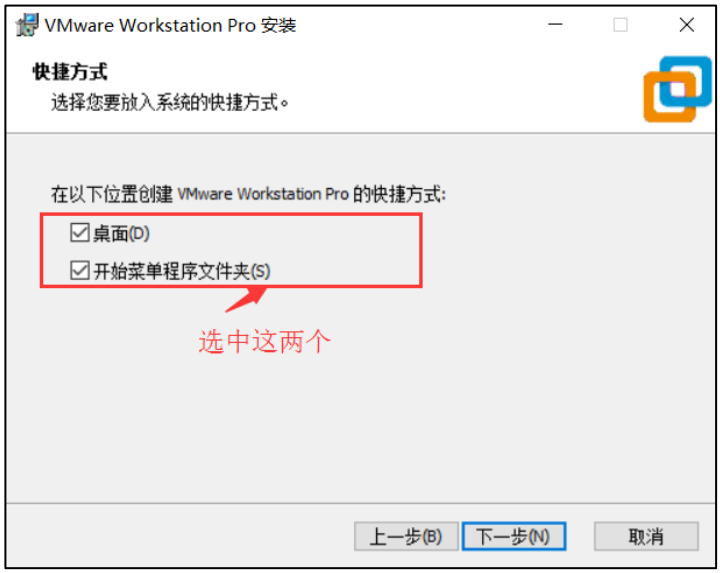



图 1.1.7 快捷方式设置

在图 1.1.7 中有两个选项，我们都选中，这样在安装完成以后就会在开始菜单和桌面上有 VMware 的图标，选中以后点击图 1.1.7 中的“下一步”，进入图 1.1.8 界面：

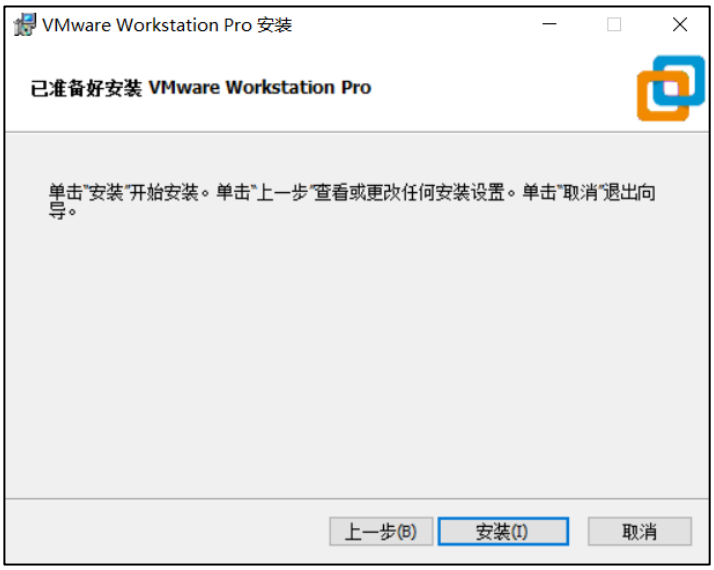



图 1.1.8 安装确定界面

前面几步已经设置好安装参数了，如果不需要修改安装参数的话就点击图 1.1.8 中的“安装”按钮开始安装 VMware，安装过程如图 1.1.9 所示：

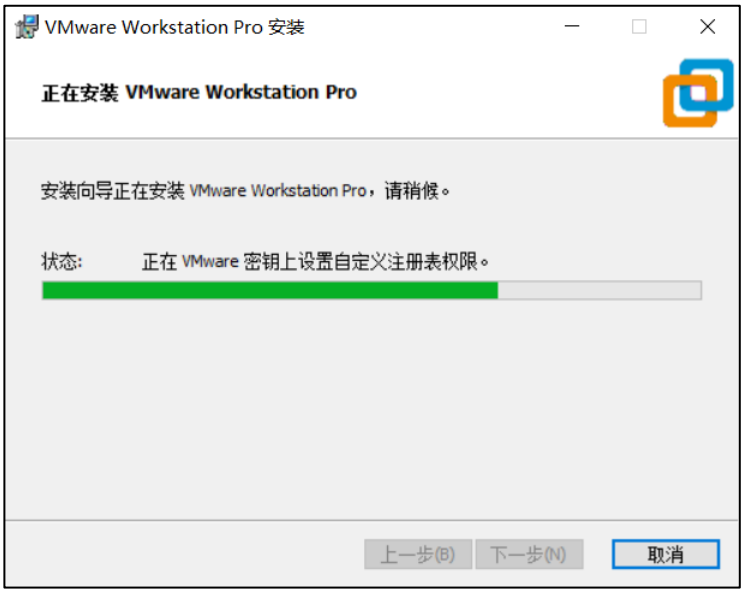



图 1.1.9 安装进行中

图 1.1.9 就是安装过程，耐心等待几分钟，等待安装完成，安装完成以后会有如图 1.1.10 所示提示：

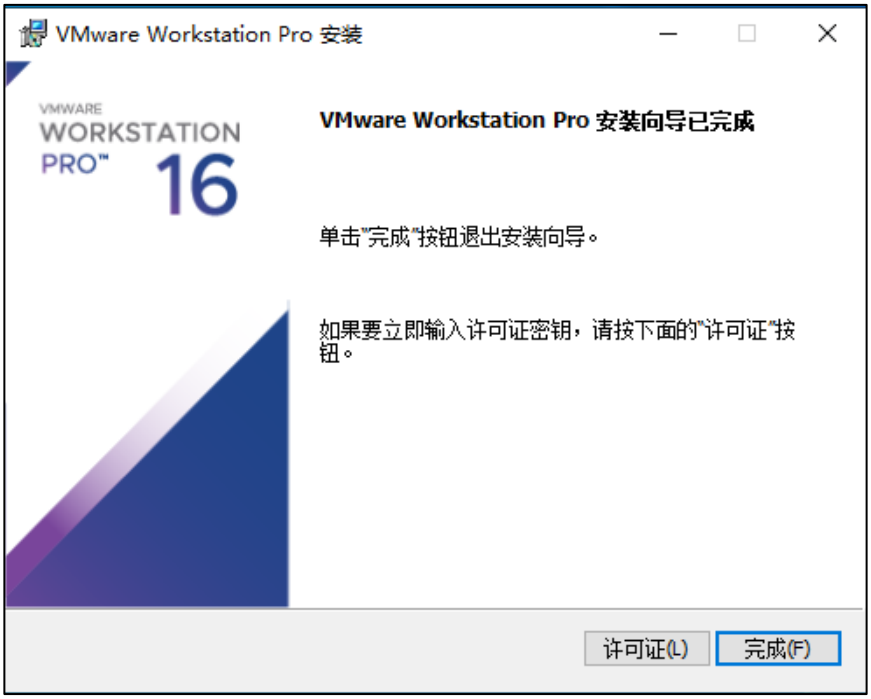



图 1.1.10 安装完成

点击图 1.1.10 中的“完成”按钮，完成 VMware 的安装，安装完成以后就会在桌面上出现 VMware Workstation Pro 的图标，如图 1.1.11 所示：

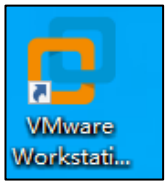



图 1.1.11 VMware 桌面图标

双击图 1.1.11 中的图标打开 VMware 软件，在第一次打开软件的时候会提示你输入许可证密钥，如图 1.1.12 所示：


图 1.1.12 输入许可证密钥

前面说了 VMware 是付费软件，是需要购买的，如果你购买了 VMware 的话就会有一串许可密钥，如果没有购买的话就选择“我希望试用 VMware Workstation 16 30 天”选项，这样你就可以体验 30 天 VMware。输入密钥以后点击“继续按钮”，如果你的密钥正确的话就会提示你购买成功，如图 1.1.13 所示：


图 1.1.13 购买 VMware 成功

点击图 1.1.13 中的“完成”按钮，VMware 软件正式打开，界面如图 1.1.14 所示：

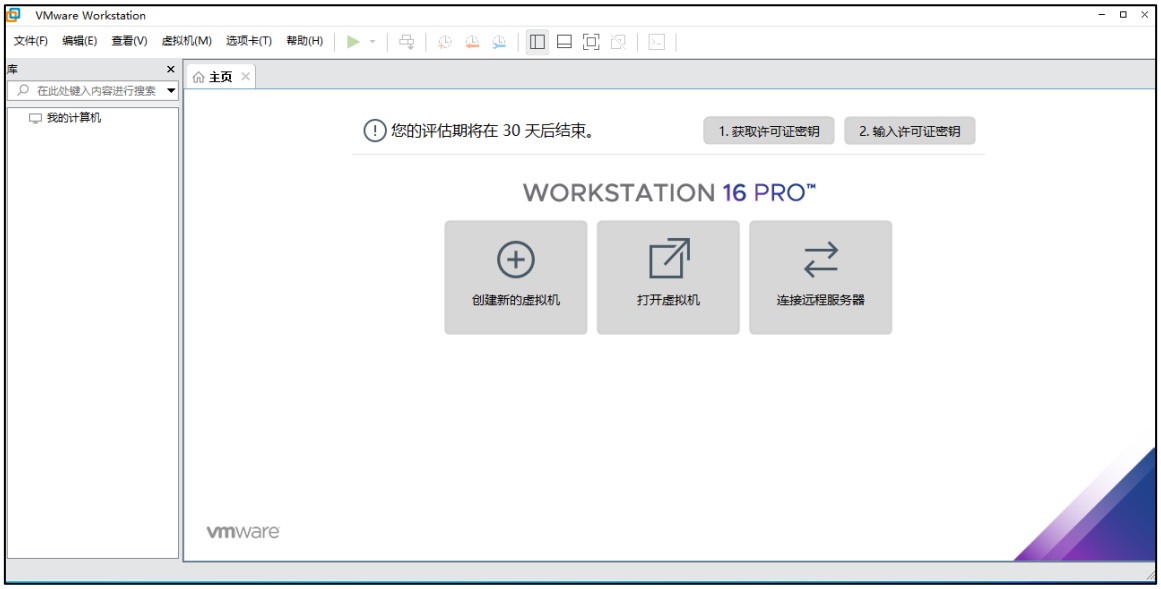



图 1.1.15 VMware Workstation 主界面

至此，虚拟机软件 VMware 安装成功。

## 1.2 创建虚拟机

安装好 VMware 以后我们就可以在 VMware 上创建一个虚拟机，打开 VMware，选择：文件->新建虚拟机，如图 1.2.1 所示：

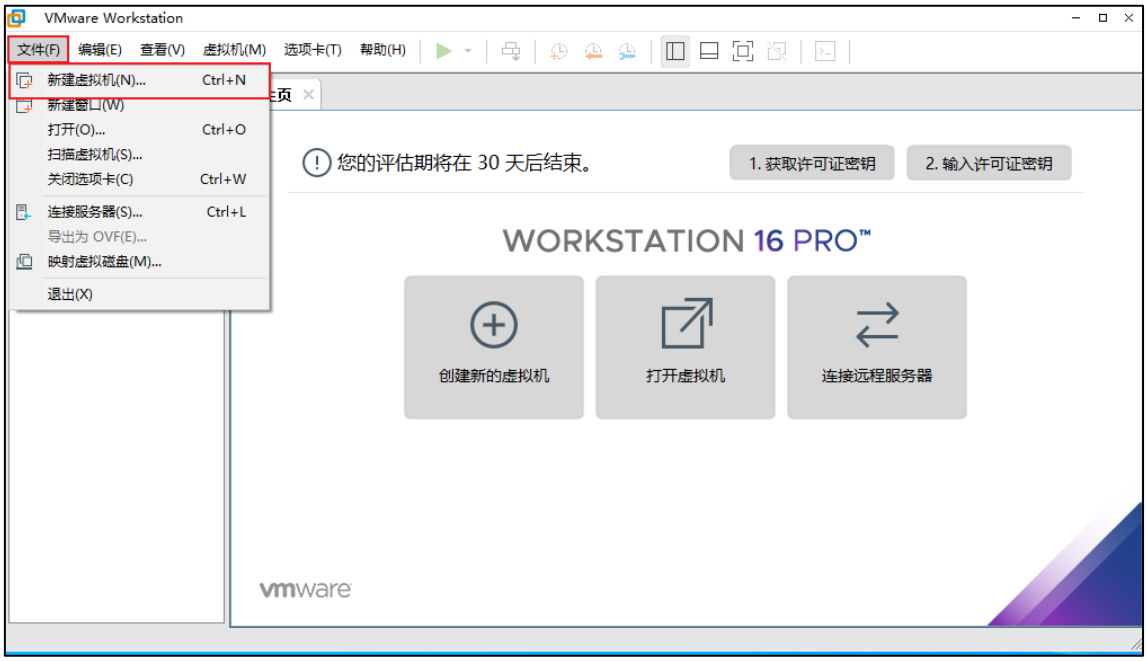



图 1.2.1 新建虚拟机

打开图 1.2.2 所示创建虚拟机向导界面：

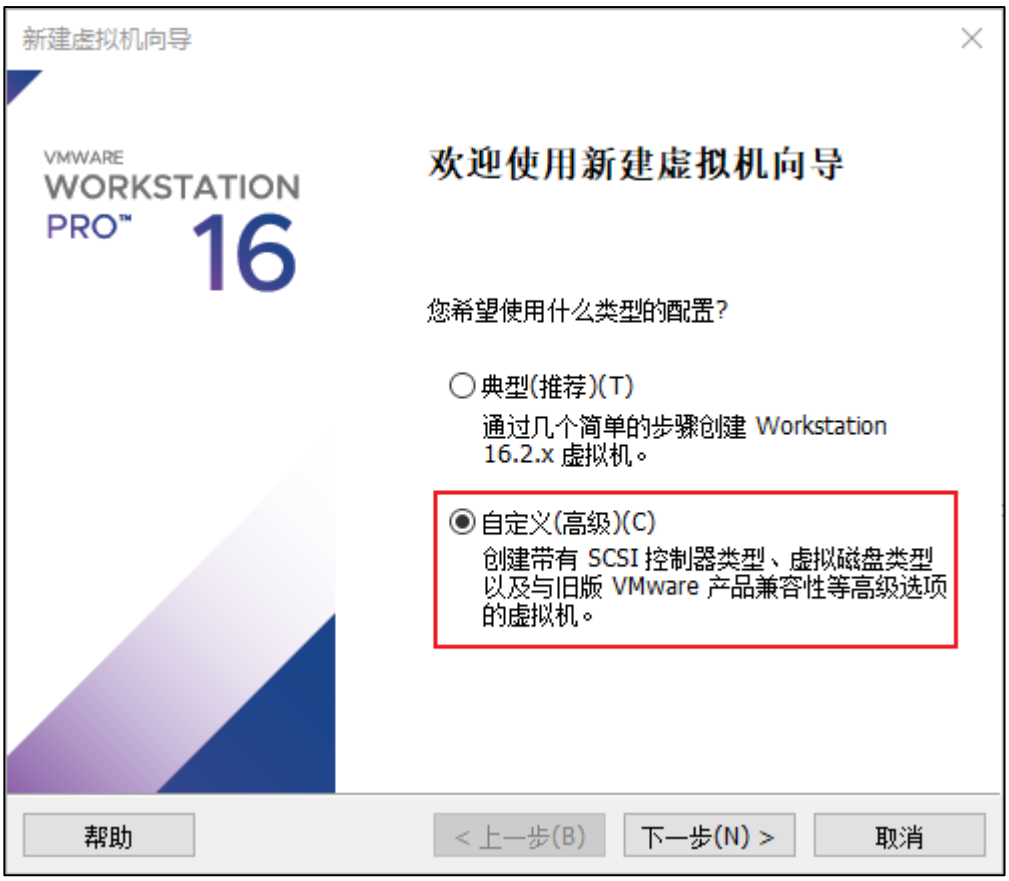



图 1.2.2 创建虚拟机向导

选中图 1.2.2 中的“自定义”选项，然后选择“下一步”，进入图 1.2.3 所示硬件兼容性选择界面：


图 1.2.3 硬件兼容性选择

在图 1.2.3 中我们使用默认值就行了，直接点击“下一步”，进入图 1.2.4 所示的操作系统安装界面：


图 1.2.4 安装客户机操作系统

图 1.2.4 就是选择你新创建的虚拟机要安装什么系统？windows 还是 linux，如果你要现在就安装系统的话需要准备好系统文件，一般是.iso 文件。我们现在不安装系统，因此选择“稍后安装操作系统(S)”这个选项，然后选择“下一步”，进入图 1.2.5 所示界面：


图 1.2.5 客户机操作系统选择

图 1.2.5 中依旧是让你选择你要在虚拟机中装什么系统，图 1.2.5 是和图 1.2.4 配合在一起使用的，在图 1.2.4 中放入系统文件(.iso 文件)，然后在图 1.2.5 中选择你放入的是什么系统，然后 VMware 就会稍后自动安装所设置的系统。在图 1.2.4 中我们没有设置系统文件，因此图 1.2.5 是没用的，不过我们还是在图 1.2.5 中的客户机操作系统一栏选择“Linux”，版本选择 Ubuntu 64 位，然后点击“下一步”，进入图 1.2.6 所示界面：

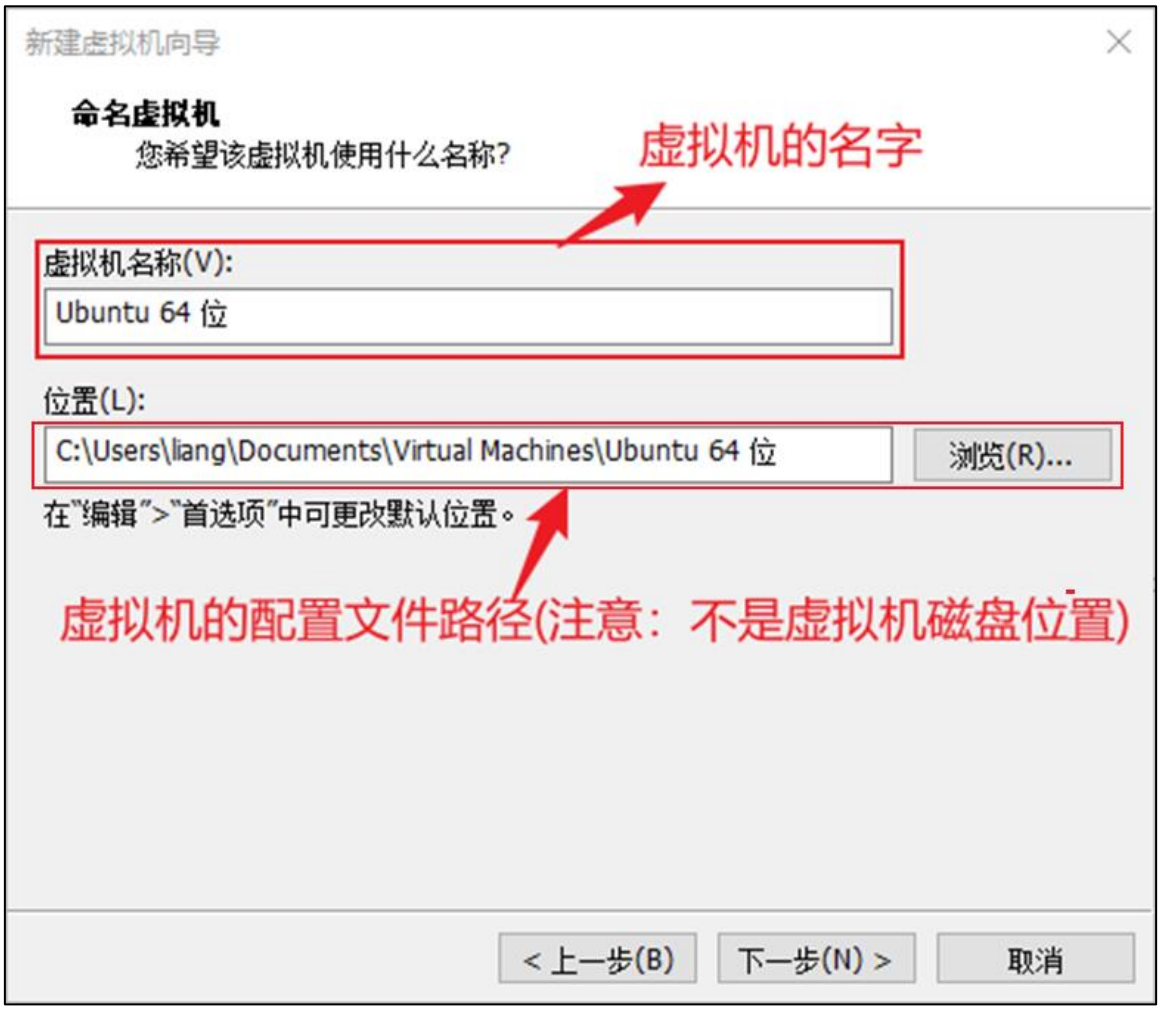



图 1.2.6 命名虚拟机

图 1.2.6 中第一个红色框设置虚拟机名字，第二个红色框里设置虚拟机的配置文件路径，大家可以根据自己的使用习惯给虚拟机命名和设置虚拟机的位置。这里笔者的虚拟机的路径如下图所示：

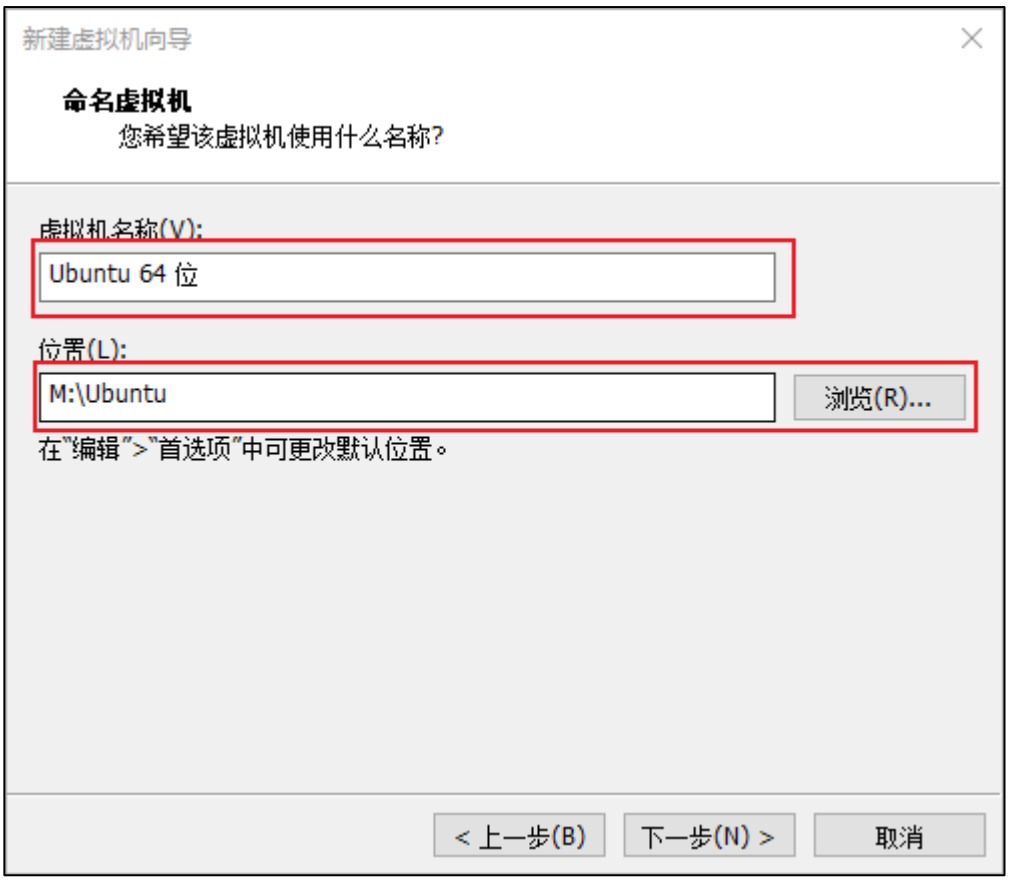



图 1.2.7 设置虚拟机的位置

设置好了，点击“下一步”，进入图 1.2.8 所示的处理器配置选择界面：


图 1.2.8 处理器配置界面

图 1.2.8 中就是配置你的虚拟机所使用的处理器数量，以及每个处理器的内核数量，这个要根据自己实际使用的电脑 CPU 配置来设置。比如我的电脑 CPU 是 I7-4720HQ，这是个 4 核 8 线程的 CPU，因此我就可以分 2 个核给 VMware，然后 I7-4720HQ 每个物理核有两个逻辑核，因此每个处理器的内核数量就是 2，所以的 VMware 虚拟机配置就如图 1.2.8 所示，大家根据自己的实际电脑 CPU 配置来设置即可，设置好以后点击“下一步”，进入图 1.2.9 所示内存配置界面：


图 1.2.9 内存配置

同样的在图 1.2.9 中根据自己电脑的实际内存配置来设置分给虚拟机的内存大小，比如我的电脑是 16GB 的内存，因此我可以给虚拟机分配 8GB 的内存(编译 ATK-DLRV1126 SDK 包需要 8GB 以上的内存)。配置好虚拟机的内存大小以后点击“下一步”，进入图 1.2.10 所示的网络类型选择界面：


图 1.2.10 网络类型选择界面

在图 1.2.10 中我们选择“使用桥接网络”，然后点击“下一步”，进入图 1.2.11 示的选择 I/O 控制器类型界面：


图 1.2.11 I/O 控制器选择

I/O 控制器类型选择默认值就行，也就是“LSI Logic”，然后点击“下一步”，进入磁盘类型选择界面，如图 1.2.12 所示：


图 1.2.12 磁盘类型选择

图 1.2.12 中选择磁盘类型，使用默认值“SCSI”即可，然后点击“下一步”，进入选择磁盘界面，如图 1.2.13 所示：

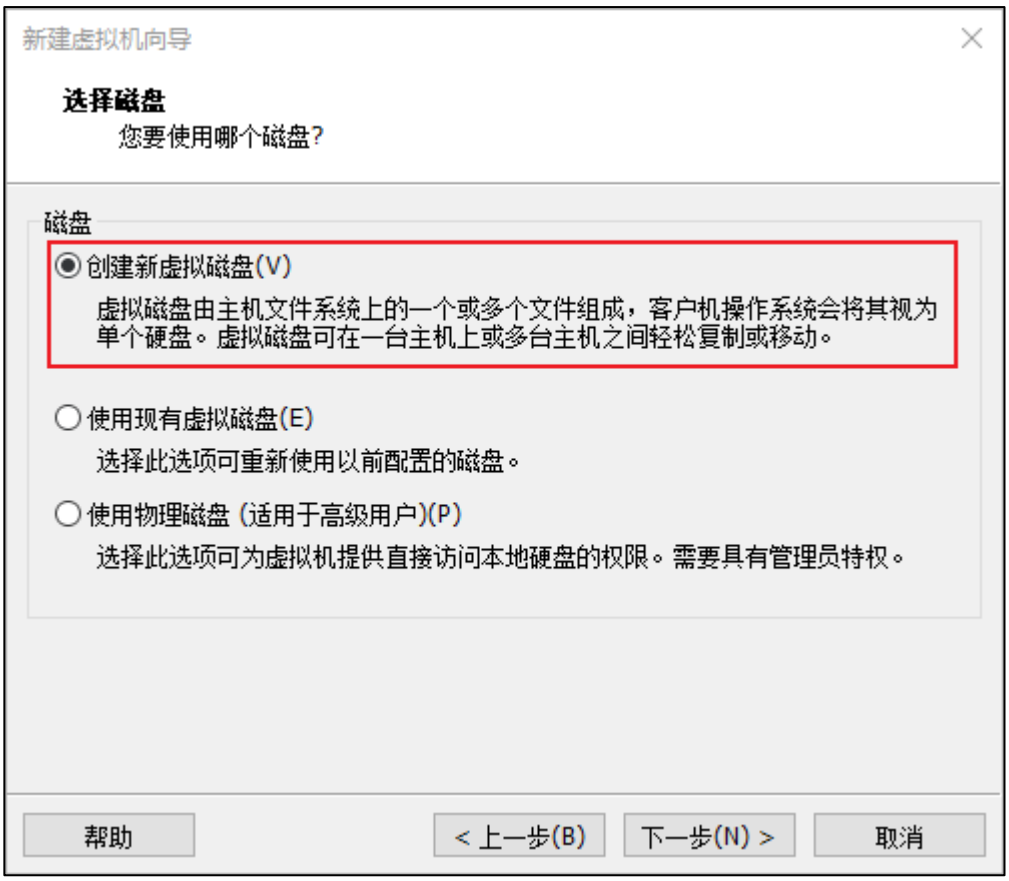



图 1.2.13 磁盘选择

图 1.2.13 中使用默认值，即“创建新虚拟磁盘”，这样我们前面设置好的那个空的磁盘就会被创建为一个新的磁盘，设置要以后点击“下一步”，进入磁盘容量设置界面，如图 1.2.14 所示：


图 1.2.14 磁盘容量设置

图 1.2.14 是用来设置虚拟机的磁盘大小，磁盘大小根据自己的电脑而定，这里笔者设置给 500GB 的空间，然后点击“下一步”，进入图 1.2.15 所示界面指定磁盘路径和名字：

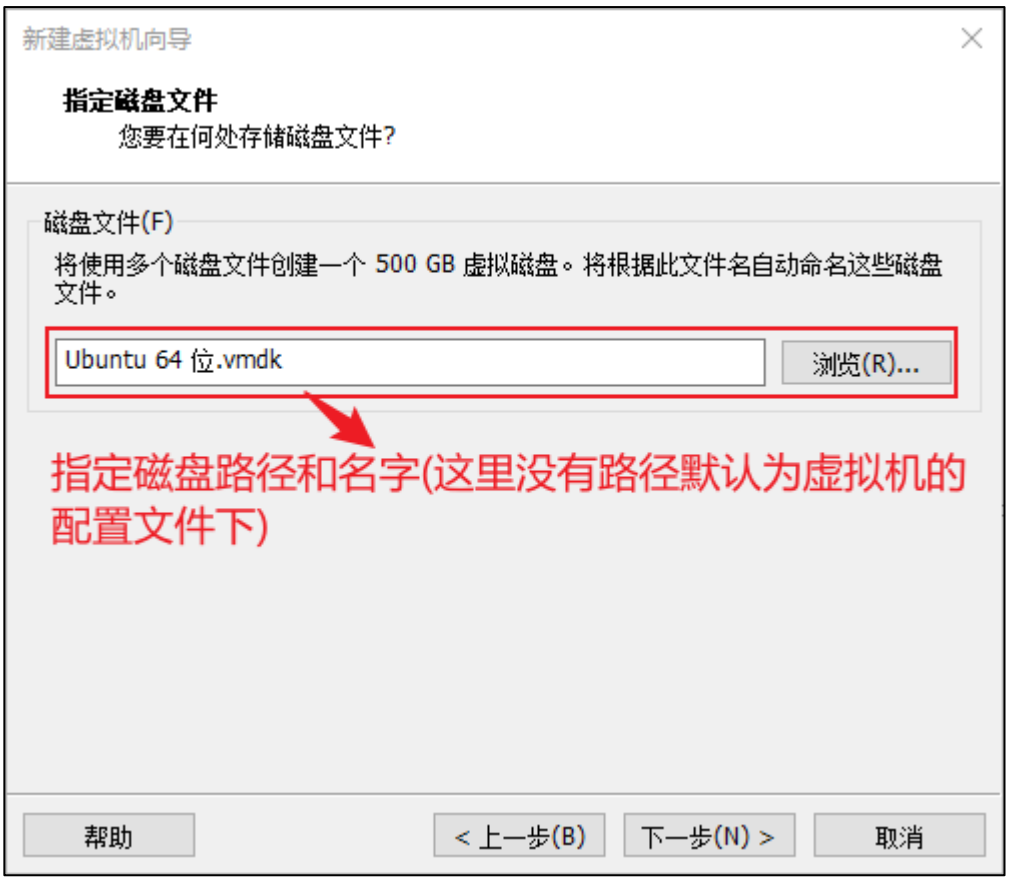



图 1.2.15 指定磁盘文件

图 1.2.15 使用默认设置，不要做任何修改，直接点击“下一步”，进入已准备好创建虚拟机界面，如图 1.2.16 所示：

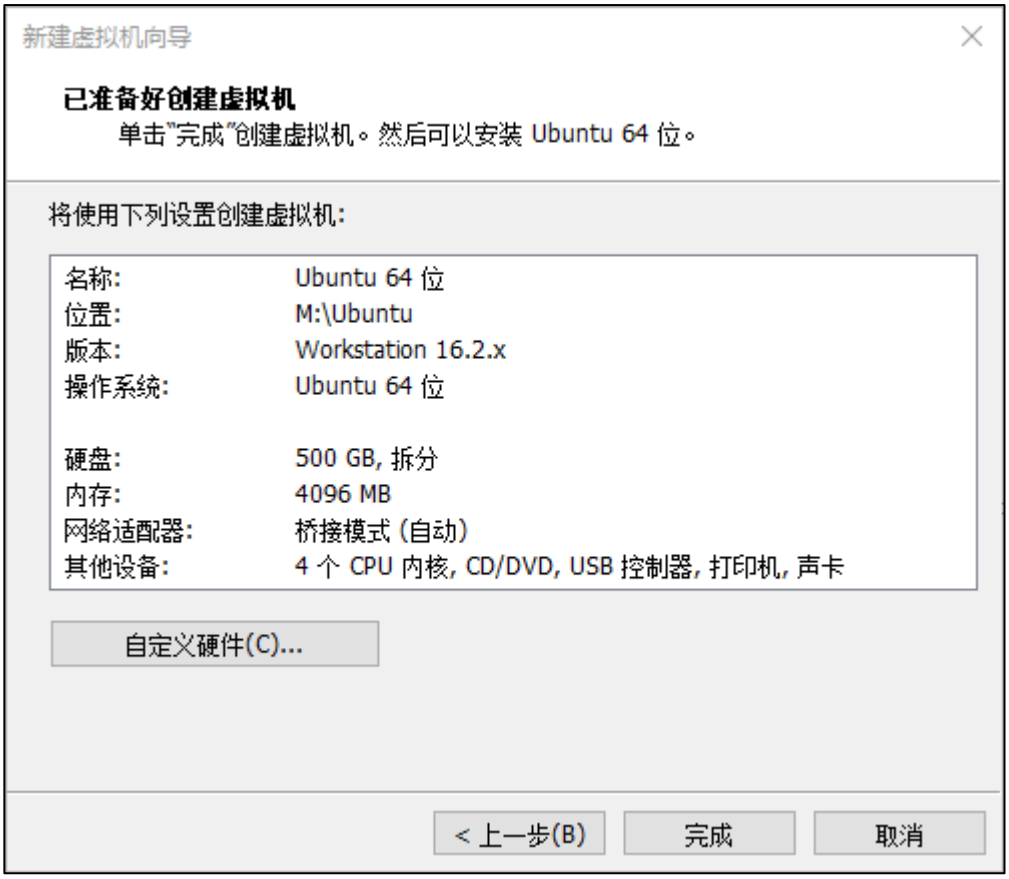



图 1.2.16 准备创建虚拟机

在图 1.2.16 中确认自己的虚拟机配置，如果确认无误就点击“完成”，如果有误的话就返回有误的配置界面做修改，点击“完成”按钮以后就会创建一个虚拟机，如图 1.2.17 所示：

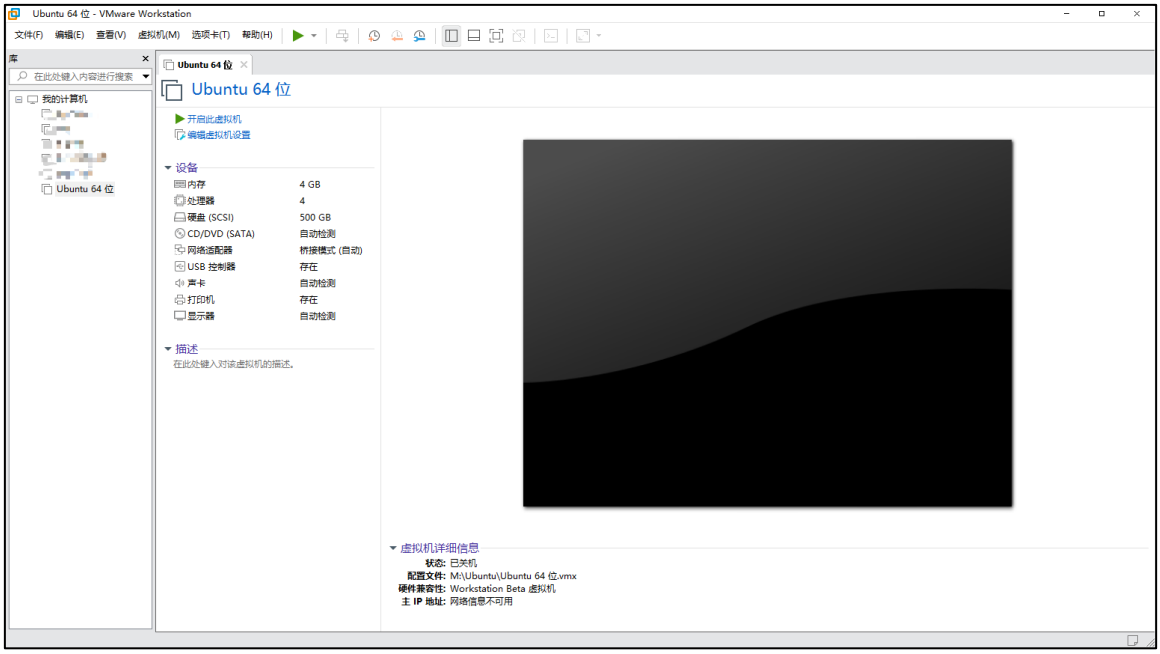



图 1.2.17 新创建的虚拟机

创建虚拟机成功以后就会在右侧的：我的计算机下出现刚刚创建的虚拟机“Ubuntu 64 位”，点击一下就会在右侧打开这个虚拟机的详细信息，如图 1.2.18 所示：


图 1.2.18 新建虚拟机配置信息

在图 1.2.18 中的设备一栏我们可以看到虚拟机详细的配置信息，图 1.2.19 所示示的两个按钮就是虚拟机的开关，

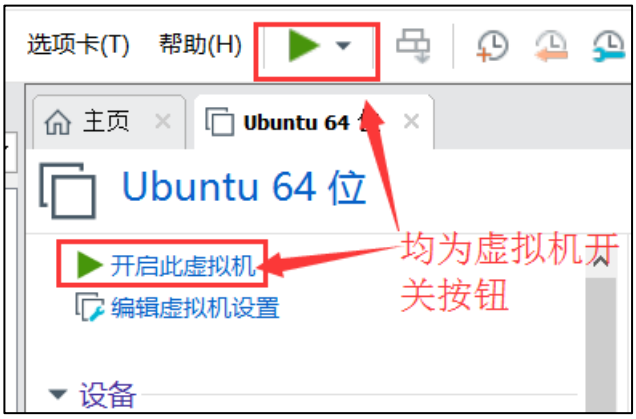



图 1.2.20 虚拟机开关机

图 1.2.20 中的这两个绿色三角按钮都可以打开虚拟机，但是此时虚拟机没有安装任何操作系统，因此没法打开，接下来我们就是要在刚刚新建的这个虚拟机中安装 Ubuntu 操作系统。

# 第二章 安装 Ubuntu 操作系统

## 2.1 获取 Ubuntu 系统

前面虚拟机已经创建成功了，相当于硬件已经准备好了，接下来就是要在虚拟机中安装 Ubuntu 系统了，首先肯定是获取到 Ubuntu 的系统镜像，Ubuntu 系统镜像肯定是在 Ubuntu 官网获取，下载地址为：https://www.ubuntu.com/download/desktop，如图 2.1.1 所示：

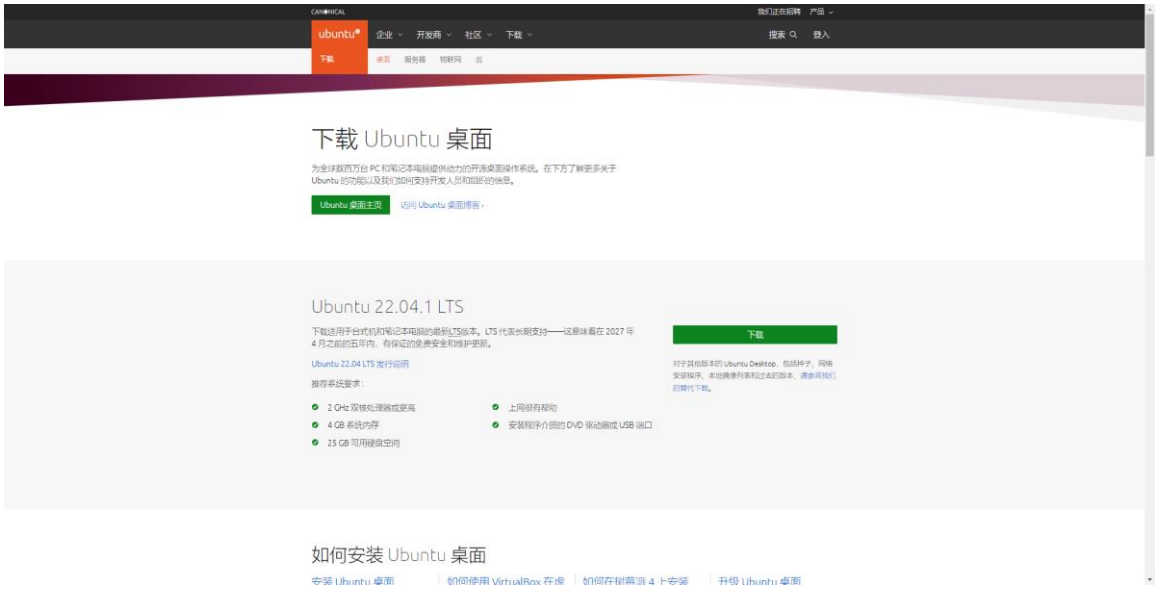



图 2.1.1 Ubuntu 最新版系统下载

从图 2.1.1 中可以看出，最新的 Ubuntu 系统为 22.04.1，此版本是官方长期支持的版本，笔者经过测试发现此版本不适合做 rv1126 开发板的开发虚拟机，Ubuntu 20 勉强可以做开发，所以我们接下来要下载和安装的是 20.04.5 版本的 Ubuntu。点击下载地址：Ubuntu20.04.5 进行下载。点击上面的链接选择 20.04.5 版本下载即可。我已经下载下来放到了开发板光盘中，路径为：开发板光盘 A-基础资料→4、软件→ ubuntu-20.04.5-desktop-amd64.iso。

## 2.2 安装 Ubuntu 操作系统

Ubuntu 系统获取到以后就可以安装了，打开 VMware 软件，选择：虚拟机->设置，如图 2.2.1 所示：

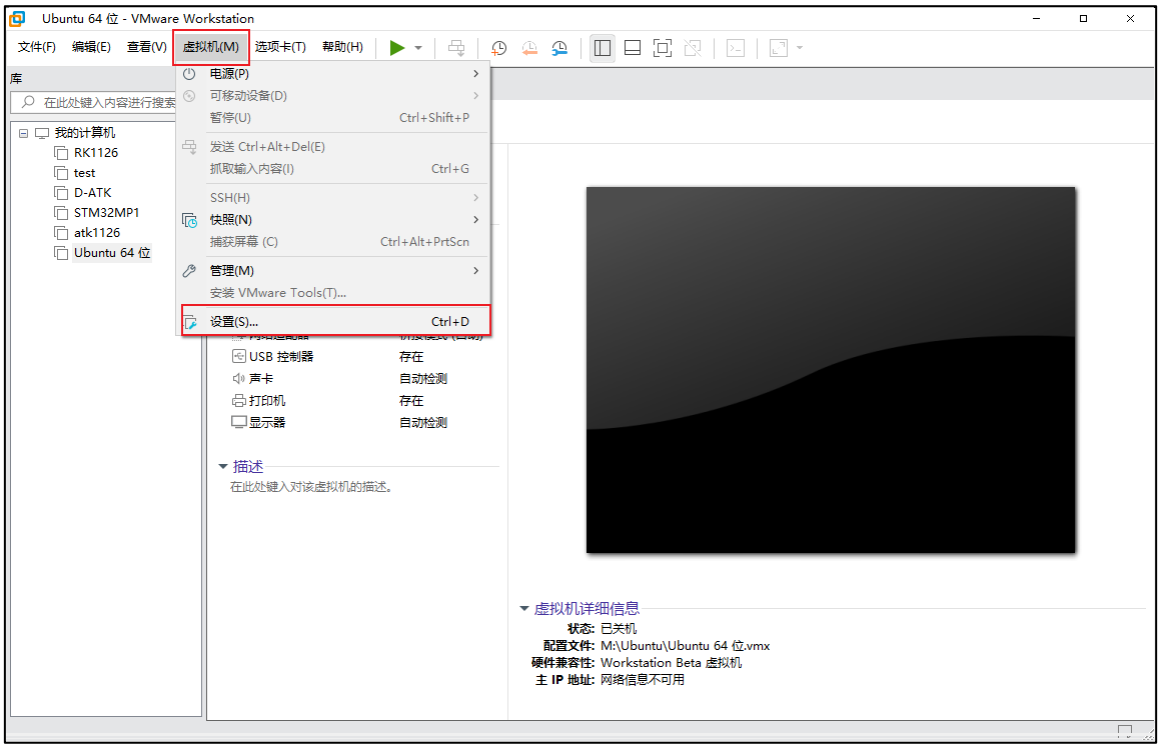


图 2.2.1 打开虚拟机设置对话框

打开以后的虚拟机设置对话框如图 2.2.2 所示：

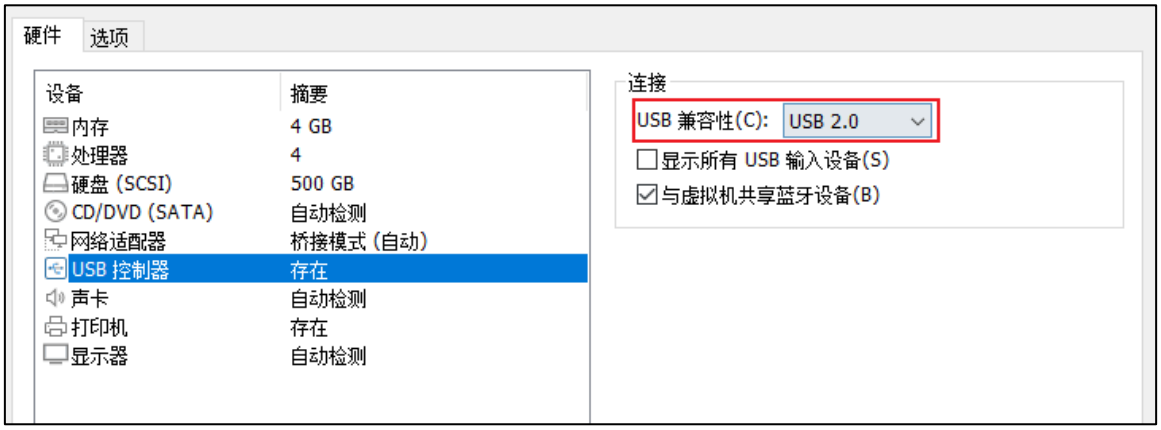



图 2.2.2 虚拟机对话框

首先设置“USB 控制器”选项，默认 USB 控制器的 USB 兼容性为 USB2.0，这样当你使用 USB3.0 的设备的时候 Ubuntu 可能识别不出来，因此我们需要调整 USB 兼容性为 USB3.0，如图 2.2.3 所示：


图 2.2.3 USB 兼容性设置

设置要 USB 兼容性以后就开始安装 Ubuntu 系统了，选中虚拟机设置对话框中的“CD/DVD(SATA)”选项，然后在右侧选中“使用 ISO 映像文件”，如图 2.2.4 所示：

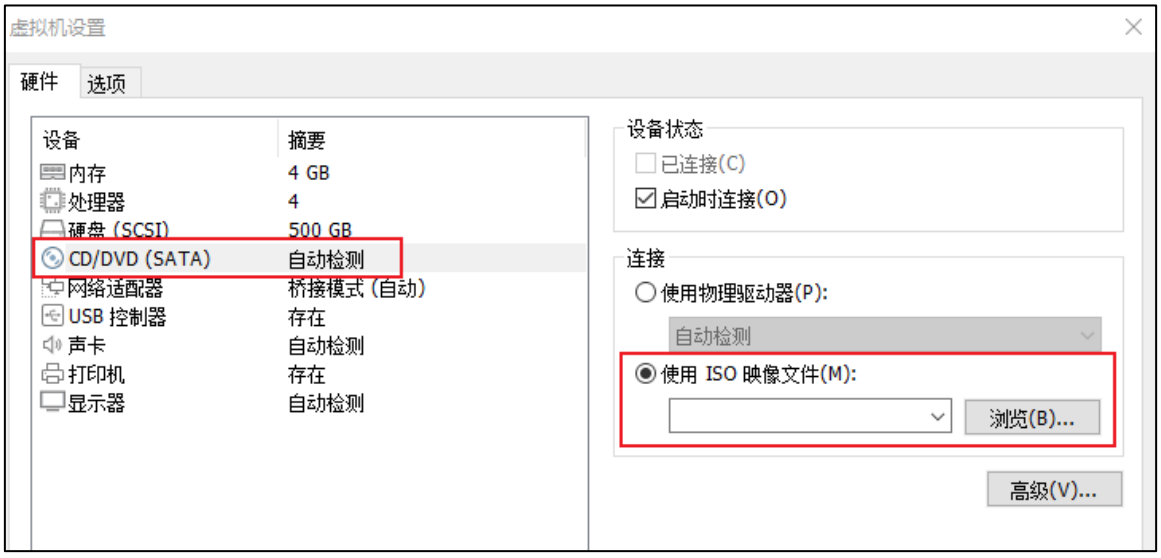



图 2.2.4 系统镜像设置

在图 2.2.4 中的“使用 ISO 映像文件”里面添加我们刚刚下载到的 Ubuntu 系统镜像，点击“浏览”按钮，选择 Ubuntu 系统镜像，完成以后如图 2.2.5 所示：


图 2.2.5 Ubuntu 镜像选择

设置好以后点击“确定”按钮退出，退出以后就可以打开虚拟机了，虚拟机就会自动的安装 Ubuntu 系统，如图 2.2.6 所示：

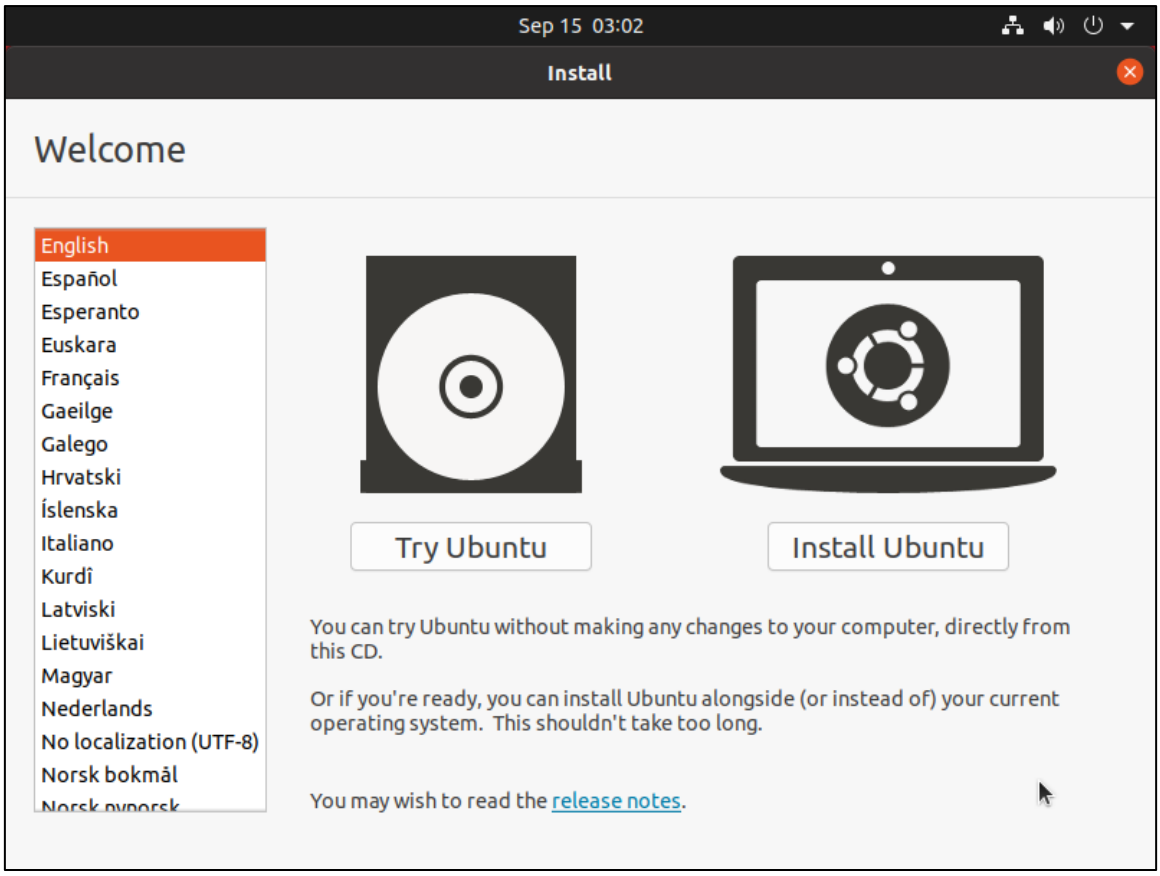



图 2.2.6 Ubuntu 安装开始

Ubuntu 开始安装以后首先是语言选择，如图 2.2.7 所示：


图 2.2.7 语言选择与安装

Ubuntu 默认语言是英文，毫无疑问，我们要选择“中文(简体)”，选择好以后点击右侧的“安装 Ubuntu”按钮，进入安装过程。安装一开始会有 7 个配置步骤，第一配置如图 2.2.8 所示，让你选择键盘布局：

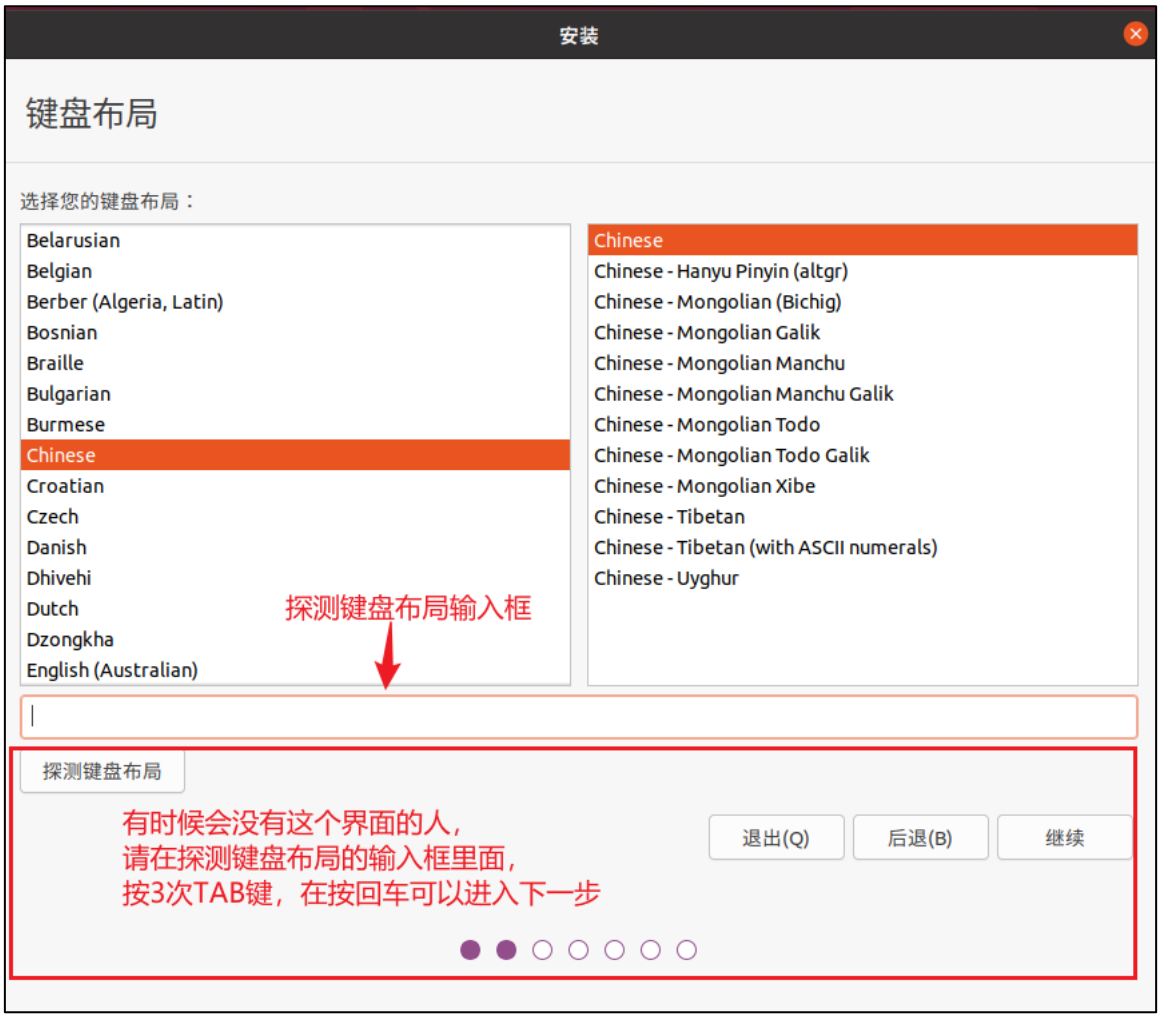



图 2.2.8 键盘布局选择

图 2.2.8 中键盘布局选择“chinese”，然后点击“继续”。接下来就是“更新和其他软件”配置界面，让你选择先安装哪些应用，是否在安装 Ubuntu 时下载更新，以及是否为图形或者无线硬件安装其它第三方软件，配置如图 2.2.9 所示：


图 2..2.9 是否安装是下载更新

直接点击图 2.2.9 中的“继续”按钮，弹出安装类型，使用默认的“清除整个磁盘并安装 Ubuntu”，如图 2.2.10 所示：


图 2.2.10 安装类型选择

设置好安装类型以后点击“现在安装”按钮，会弹出“将改动写入磁盘吗？”对话框，点击“继续”即可，下一步会让你输入你在哪个位置，输入自己所在的城市即可，比如我在广州就输入“guangzhou”，如图 2.2.11 所示：

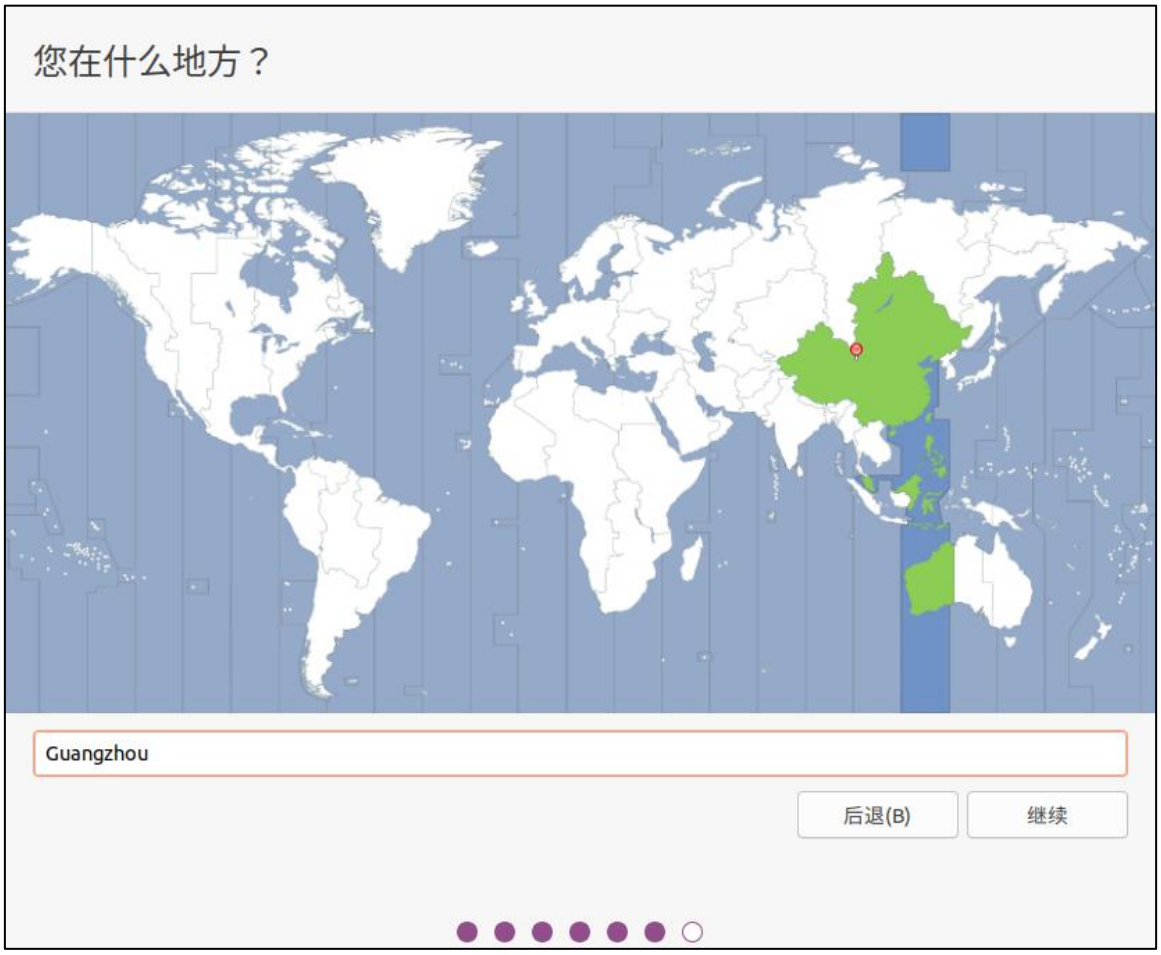



图 2.2.11 输入所在位置

输入地址以后点击“继续”按钮，进入下一步设置用户名和密码，自己设置自己的用户名和密码，比如我的设置如图 2.2.12 所示：

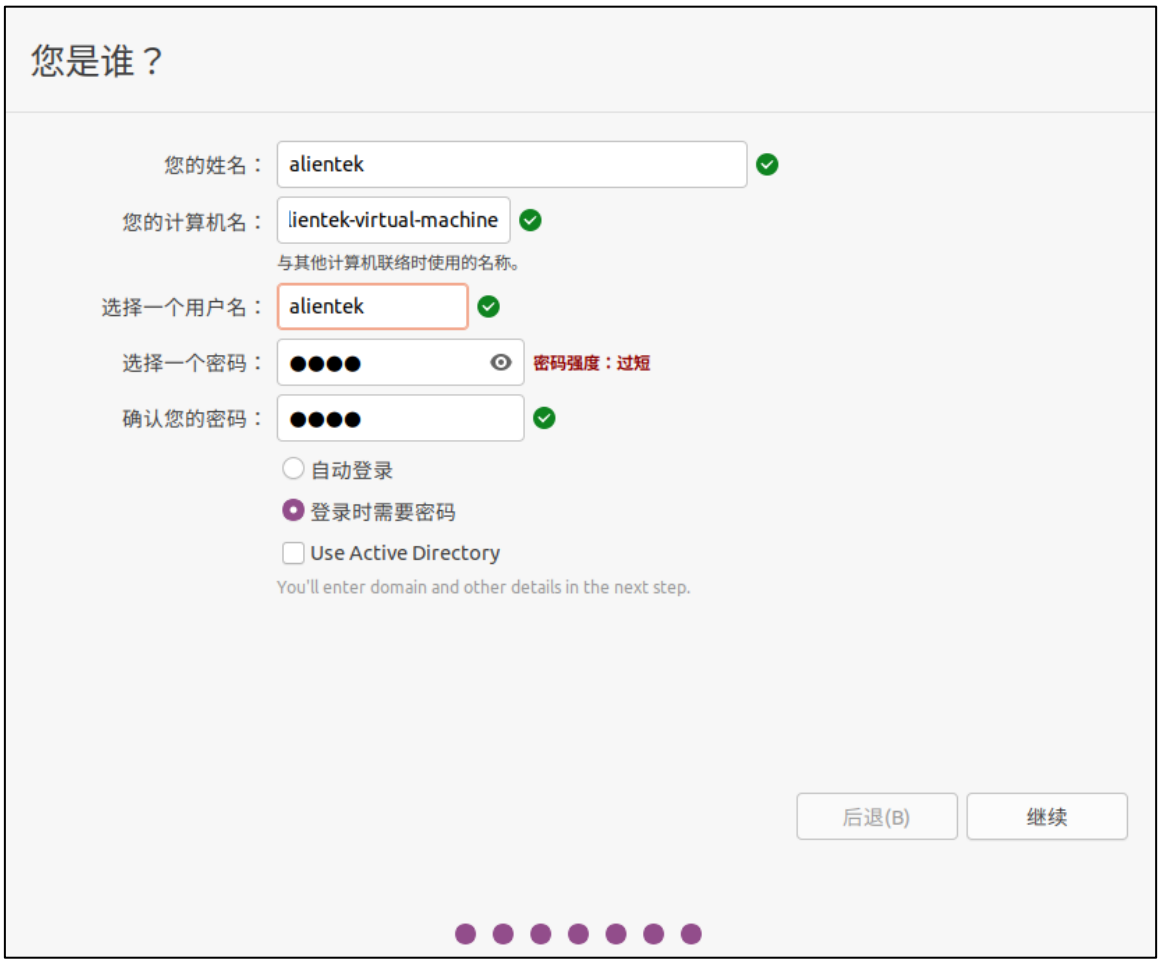



图 2.2.12 设置用户名和密码

设置好用户名和密码(密码强度提示可以不用管)以后点击“继续”按钮，系统就会开始正式安装，如图 2.2.13 所示：

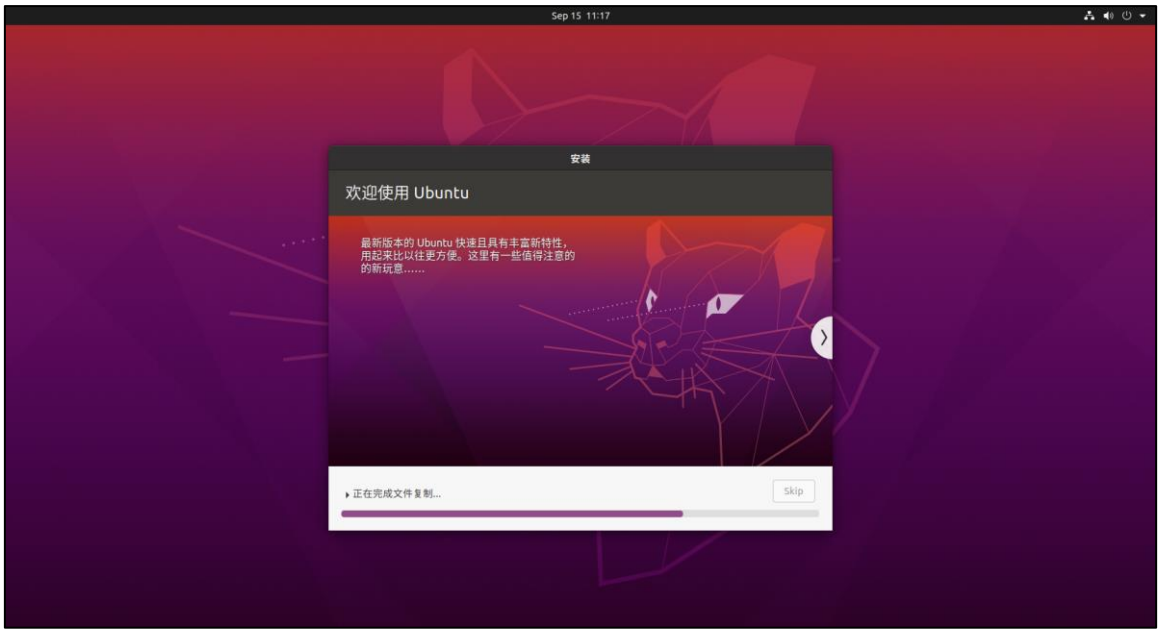



图 2.2.13 系统安装中

等待系统安装完成，安装过程中会下载一些文件，所以一定要保证电脑能够正常上网，如果不能正常上网的话可以点击右侧的“skip”按钮来跳过下载文件这个步骤，对于系统的安装没有任何影响，安装完成以后提示重启系统，如图 2.2.14 所示：

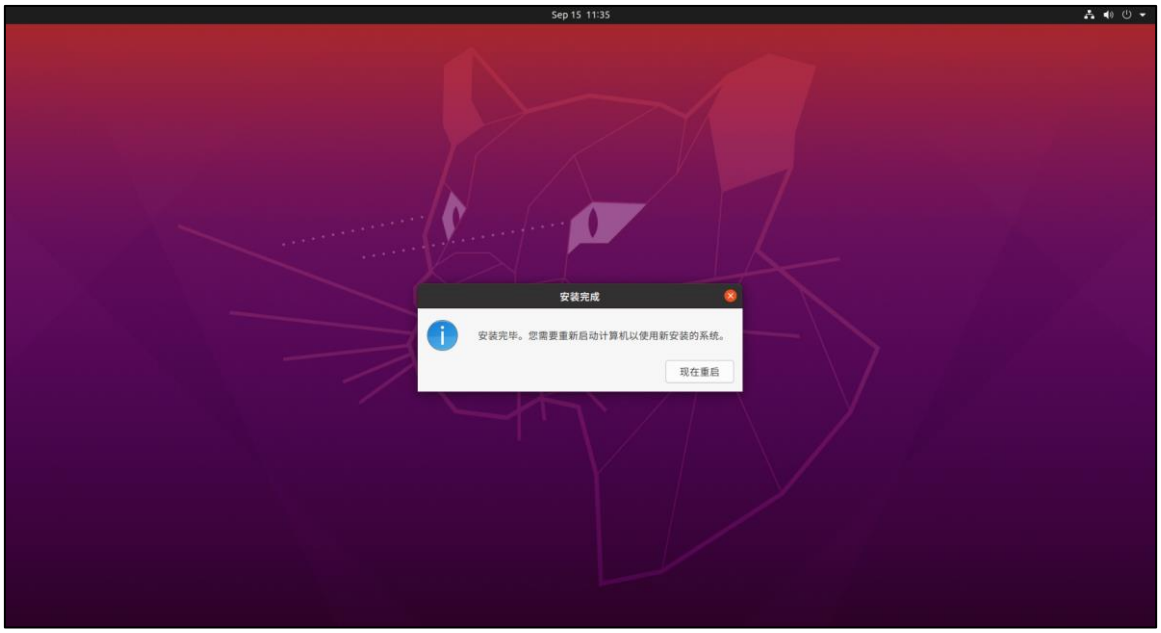



图 2.2.14 安装完成，重启系统

重启的时候需要弹出虚拟机里面的系统镜像！关闭 Ubuntu 操作系统，打开 VMware 的虚拟机设置界面，然后选中“CD/DVD(SATA)”，右侧的“连接”选择“使用物理驱动器”，如图 1.3.2.15 所示。

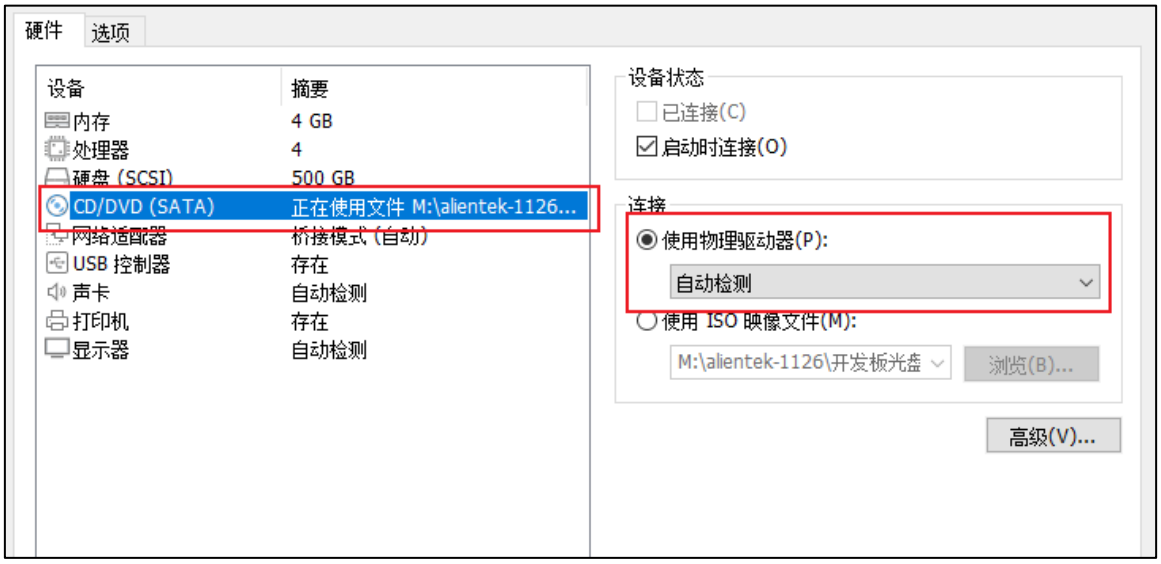



图 2.2.15 弹出 Ubuntu 系统镜像

设置好以后点击“确定”按钮，然后重新打开虚拟机，启动 Ubuntu，系统启动以后就会提示输入密码，如图 2.2.16 所示：

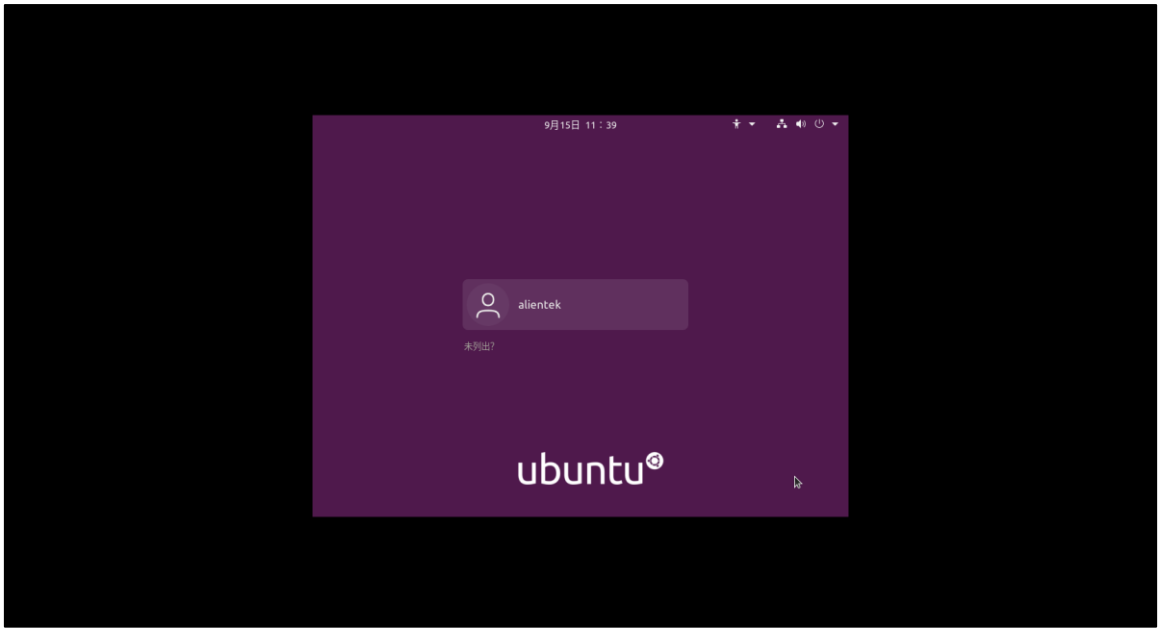



图 2.2.16 系统登录界面

在图 2.2.16 中输入密码，点击键盘上的回车就会进入系统主界面，系统界面如图 2.2.17 所示：


图 2.2.17 系统桌面

图 2.2.17 就是第一次进入系统的系统桌面，第一次进入系统会有“在线账户的界面”，也就是给 Ubuntu 系统的简易引导指南，点击图 2.2.17 右上角的“前进(N)”来一步步观看指南。最

终的系统主界面如图 2.2.18 所示：

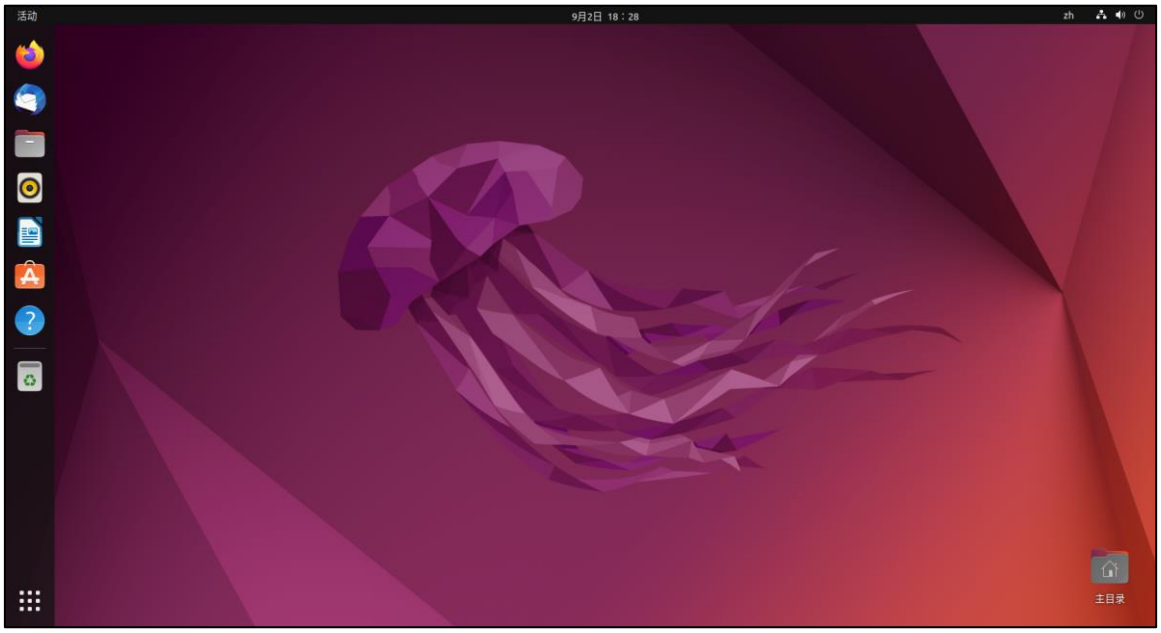



图 2.2.18 Ubuntu 系统主界面

## 2.3 VMware tools 安装

注意：新版本的 VMware 会自动安装的 VMware tools，如何测试 VMware tools 呢？在 Windows 系统里复制一段话，能粘贴到 Ubuntu 系统里，说明 VMware tools 已经安装了。没有安装的请参考本小节。

打开虚拟机中的 Ubuntu 系统，然后点击：Vmware 的虚拟机(M)->安装 VMware Tools(T)……，如图 2.3.1 所示：


图 2.3.1 安装“VMware Tools”

点击图 2.3.1 中的“安装 WMware Tools”，此时就会在 Ubuntu 系统中自动下载 VMware Tools

工具，下载完成以后会放到 Ubuntu 桌面上，如图 2.3.2 所示：

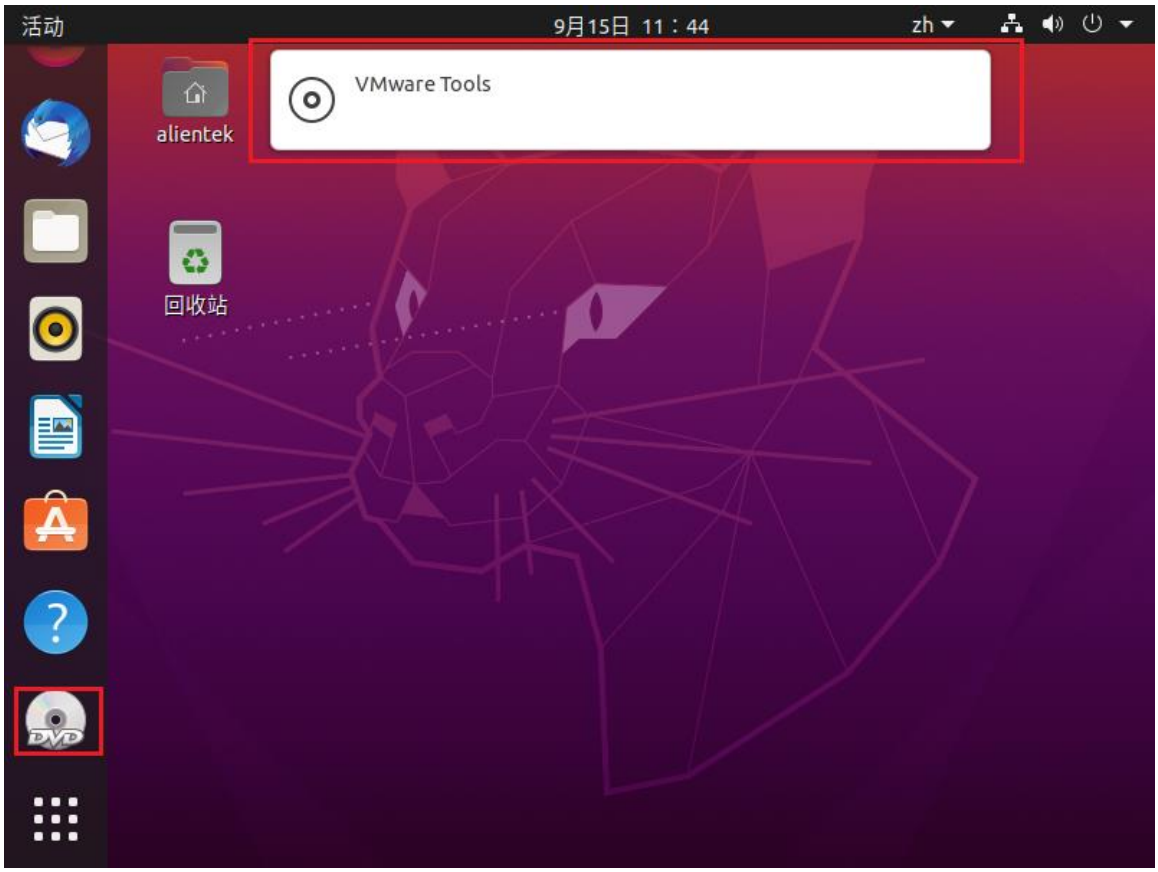



图 2.3.2 下载得到的 VMware Tools

双击图 2.3.2 中的光盘磁盘，打开以后如图 2.3.3 所示：

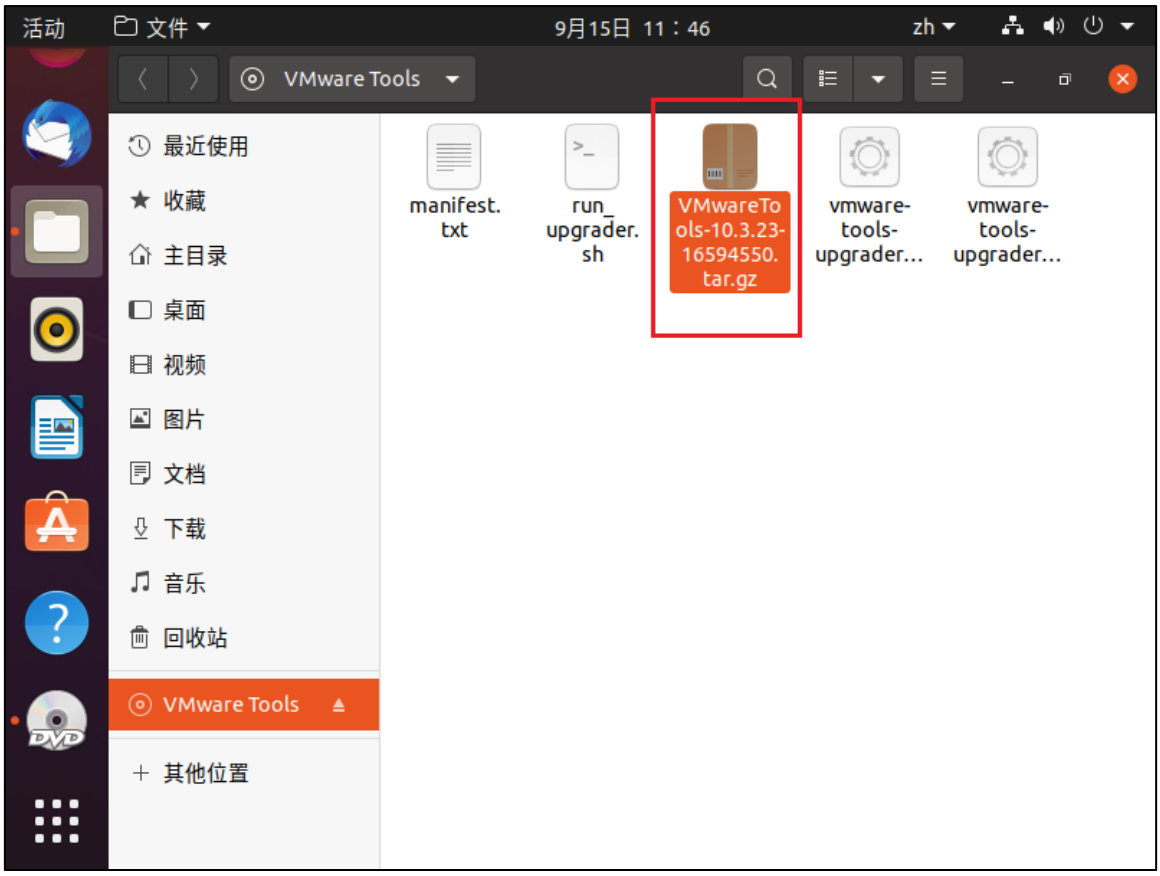



图 2.3.3 VMware Tools 安装包文件

图 2.3.3 中的 VMwareTools-10.3.23-16594550.tar.gz 就是我们要的安装包，将其解压到桌面上，如图 2.3.4 所示：

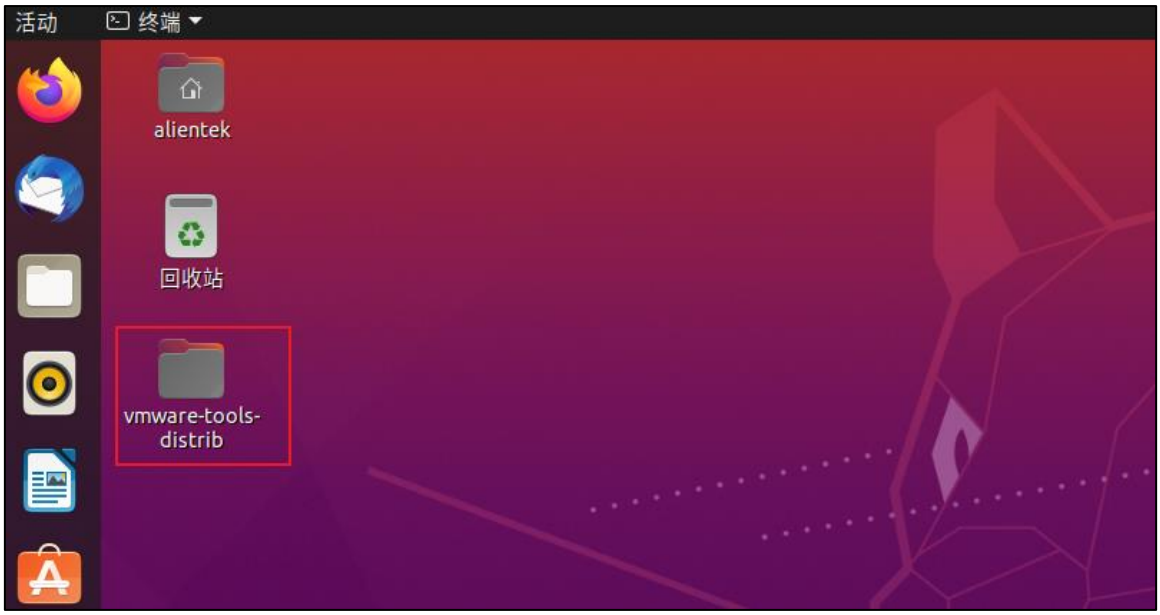



图 2.3.4 解压后的 VMwareTools

图 2.3.4 就是解压出来的 VMware Tools 安装压缩包，用终端进入此目录下，运行以下命令：

```
sudo ./vmware-install.pl                    //执行安装软件
```

安装的过程中会弹出一系列的操作提示，如果遇到选择“是”或“否”的这样的问题，输入“yes”表示是，输入“no”或者直接按回车表示“否”。如果是哪些安装路径的问题直接按回车键，使用默认路径即可，其他类型的问题大家根据实际提示做选择。安装完成以后如图 2.3.5 所示：

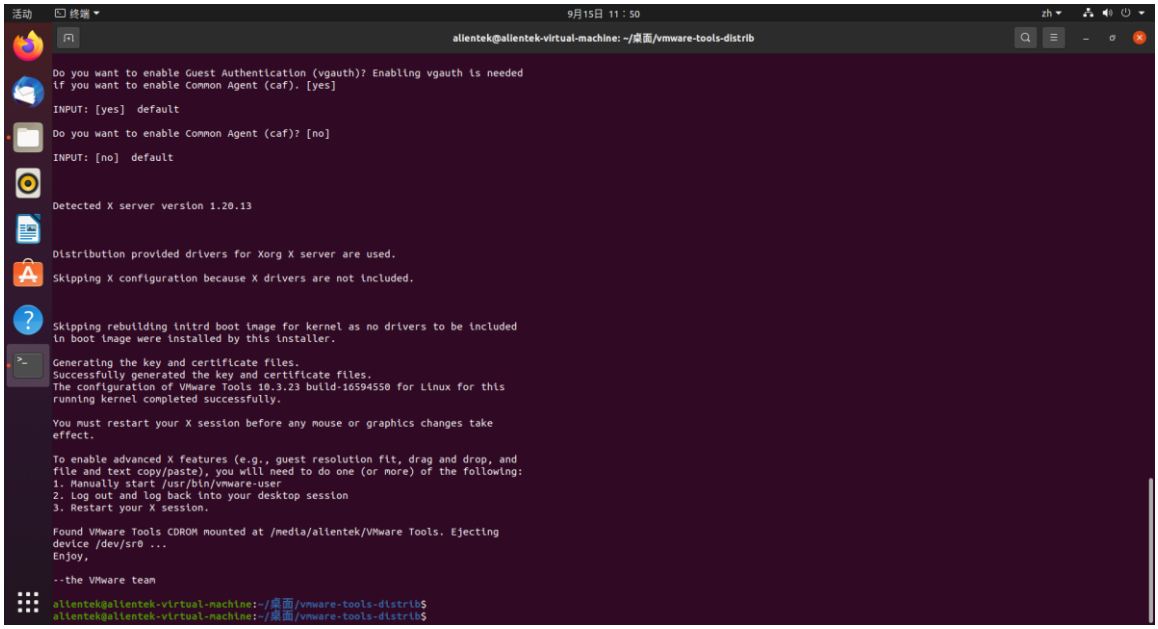



图 2.3.5 VMware Tools 安装完成

VMware Tools 安装完成以后重启 Ubuntu，重启以后就可以直接在虚拟机 Ubuntu 系统和主机 Windows 下进行文字、文件等的复制粘贴。如果还是不能复制的请运行以下命令：

```
sudo apt-get autoremove open-vm-tools//卸载已有的工具
sudo apt-get install open-vm-tools        //安装工具 open-vm-tools
sudo apt-get install open-vm-tools-desktop    //安装 open-vm-tools-desktop
```

接着重启系统即可。

# 第三章 RV1126 开发环境搭建

## 3.1 rv1126 的环境配置

在上章节里面我们已经安装好 Ubuntu，此时的 Ubuntu 还是不能做开发的，因为还有很多环境和软件没有安装，所以要先安装环境，这里笔者已经把所有坑都填完了。跟着下面一步一步走就行了。

- 先设置 Ubuntu 的源

国内的环境下使用 Ubuntu 官方的默认源是不能配置出 RV1126 的开发环境，有一些包不能安装，所以我们要设置合适的源。Ubuntu 官方更换源有一个很智能的操作，可以根据自己的网络位置设配合适的源，设置如下步骤：

打开设置，在左边设置栏里面找到“关于”如下图所示：

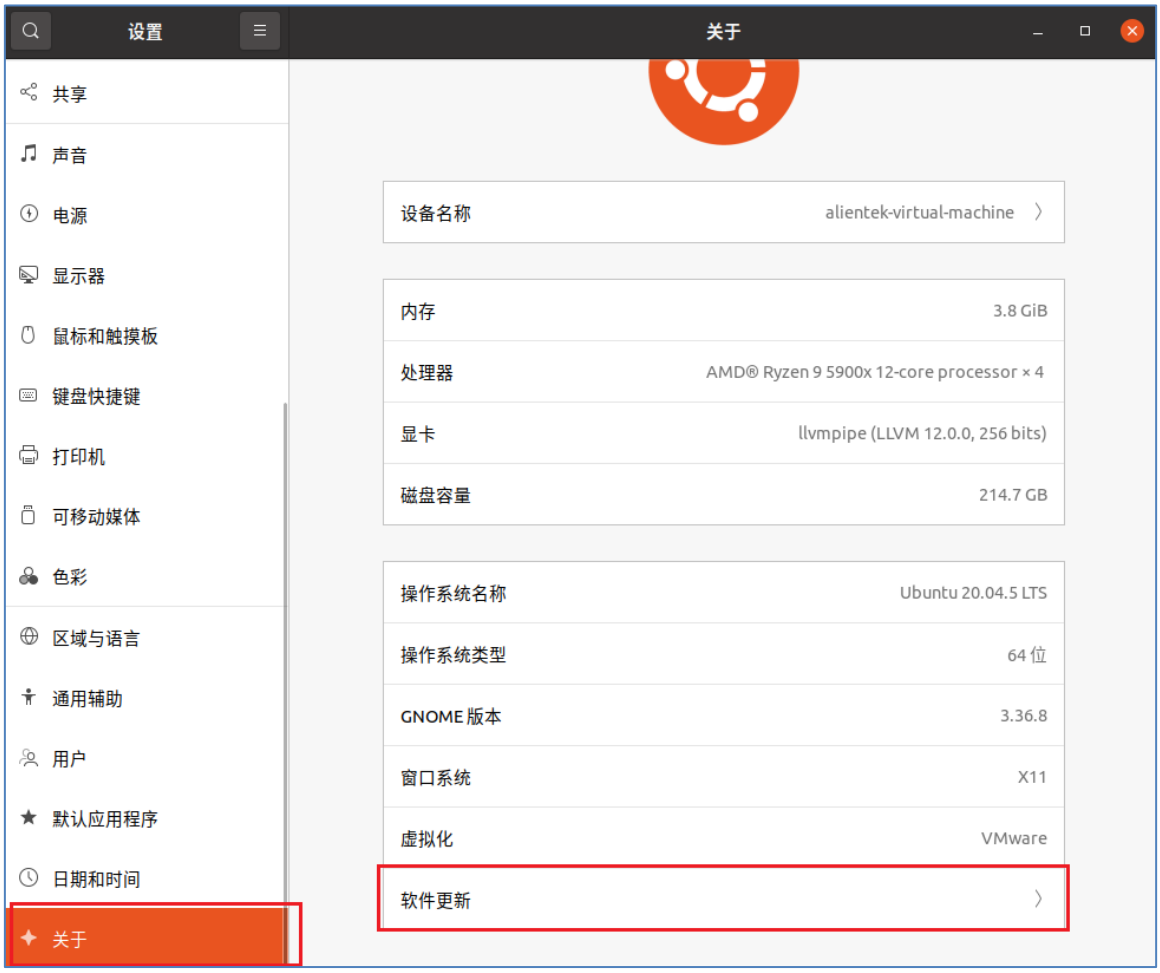



图 3.1.1 关于设置

我们找到“关于”设置后，右边栏最后面有一个软件更新设置，接下来点击“软件设置”，弹出如下界面：


图 3.1.2 软件更新图

图 3.1.2 可以看出是下载软件是在“位于中国的服务器”，此时的源链接还是不能配置 RV1126 开发环境，展开红色框里面，选择“其它”如下图所示：

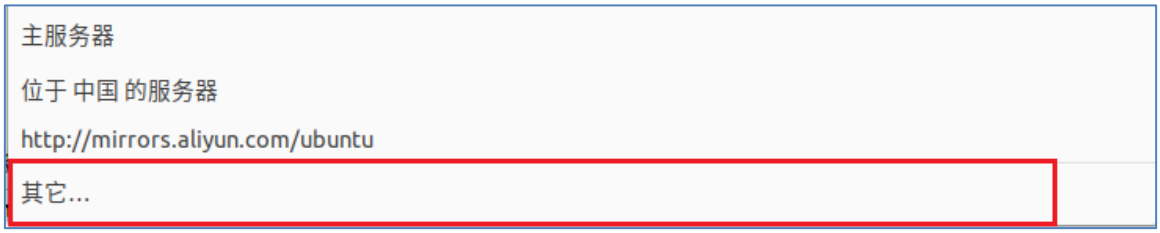



图 3.1.3 源服务器选择

点击成功就会出现如下图所示：

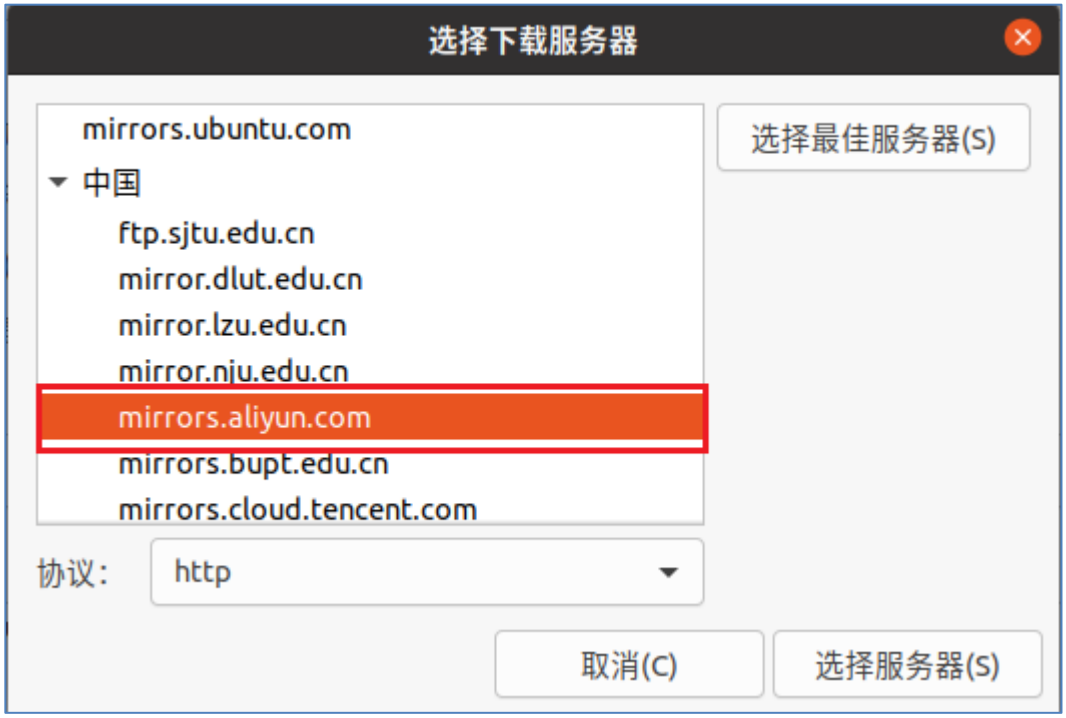



图 3.1.4 选择下载服务器

从图 3.1.4 中可以看出来，有很多服务器的选择，在国内最好选择阿里源最合适。选择 mirrosr.aliyun.com 为阿里源。我们也可以根据自己的网络匹配最合适的源点击“选择最佳服务器”就可以根据网络合适的源。输入密码就可以直接更新源。最后还要更新缓存，直接点击更新即可。更新缓存的还要另一种方法用命令更新，命令如下：

```
sudo apt update
sudo apt upgrade
```

- SDK 编译环境搭建所依赖的软件包

  安装的命令如下所示：

```
sudo apt-get install  device-tree-compiler git-core u-boot-tools mtools parted libudev-dev
sudo apt-get install libusb-1.0-0-dev autoconf autotools-dev libsigsegv2 m4 intltool libdrm-dev
sudo apt-get install curl sed make binutils build-essential gcc g++ bash patch gzip gawk bzip2
sudo apt-get install perl tar cpio python unzip rsync file bc wget libncurses5 libglib2.0-dev
sudo apt-get install libgtk2.0-dev libglade2-dev cvs git mercurial openssh-client subversion
sudo apt-get install asciidoc w3m dblatex graphviz  libc6:i386 libssl-dev expect fakeroot cmake
sudo apt-get install flex bison liblz4-tool libtool keychain net-tools adb lib32gcc-7-dev g++-7
sudo apt-get install libstdc++-7-dev libncurses5-dev libncursesw5-dev openssh-server
```

注意：本来是写成一行命令安装的，想到有些人的 PDF 会自带换行符，所以就分成 8 条命令进行安装。

## 3.2 Ubuntu 和 Windows 文件互传

在开发的过程中会频繁的在 Windows 和 Ubuntu 下进行文件传输，比如在 Windwos 下进行代码编写，然后将编写好的代码拿到 Ubuntu 下进行编译。Windows 和 Ubuntu 下的文件互传我们需要使用 FTP 服务(安装 vmware tools 后可以直接拷贝文件，这种方法不推荐使用)，设置方

法如下：

**1、开启 Ubuntu 下的 FTP 服务**

打开 Ubuntu 的终端窗口，然后执行如下命令来安装 FTP 服务：

```
sudo apt-get install vsftpd
```

等待软件自动安装，安装完成以后使用如下 VI 命令打开/etc/vsftpd.conf，命令如下：

```
sudo vi /etc/vsftpd.conf
```

打开以后 vsftpd.conf 文件以后找到如下两行：

```
local_enable=YES
write_enable=YES
```

确保上面两行前面没有“#”，有的话就取消掉，完成以后如图 2.5.1 所示：

```
27 # Uncomment this to allow local users to log in.
28 local_enable=YES
29 #
30 # Uncomment this to enable any form of FTP write command.
31 write_enable=YES
```

图 3.2.1 vsftpd.conf 修改

修改完 vsftpd.conf 以后保存退出，使用如下命令重启 FTP 服务：

```
sudo /etc/init.d/vsftpd restart
```

**2、Windows 下 FTP 客户端安装**

Windows 下 FTP 客户端我们使用 FileZilla，这是个免费的 FTP 客户端软件，可以在 FileZilla 官网下载，下载地址如下：https://www.filezilla.cn/download/client，下载界面如图 2.5.2 所示：

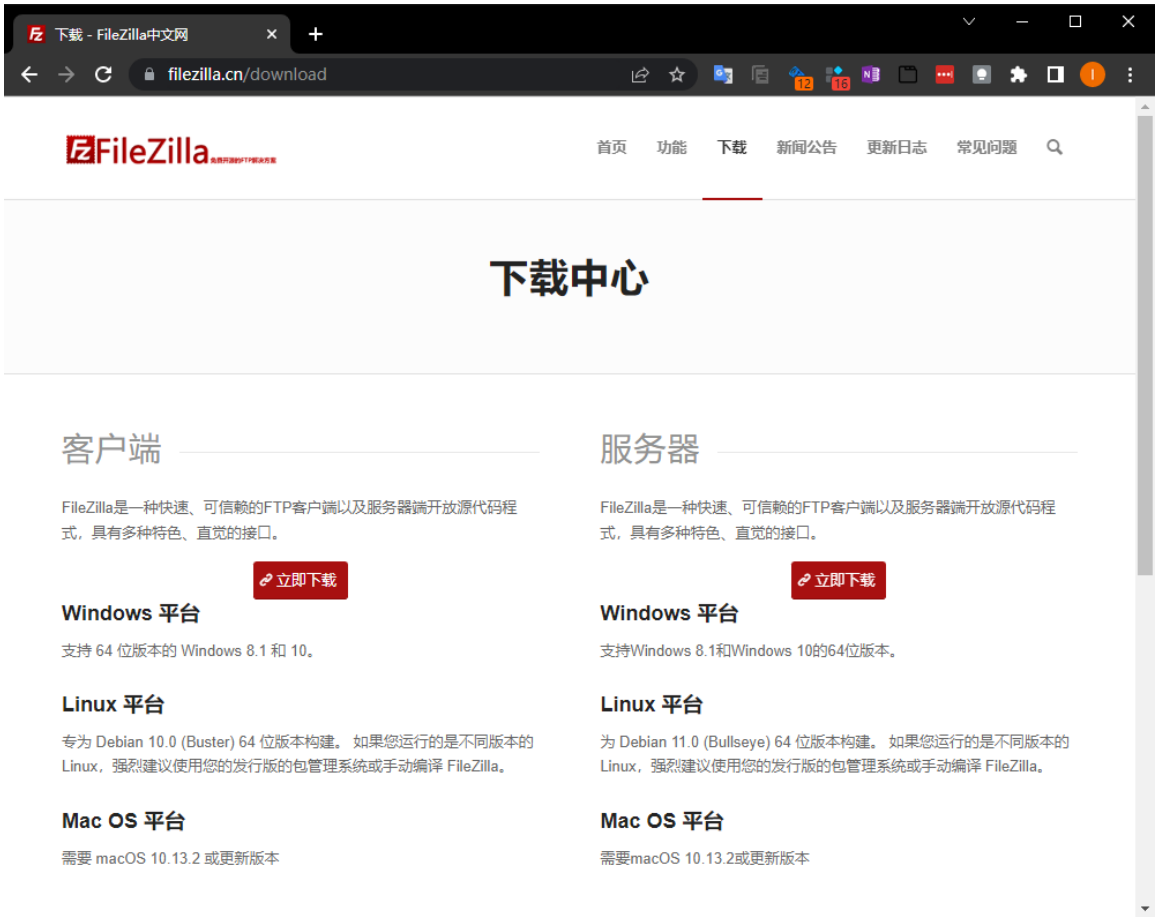


图 3.2.2 FileZilla 软件下载

如果是 32 位电脑就选择 32 位版本，64 位电脑就选择 64 位版本，我们已经下载好了 64 位版本的 FileZilla 并放到开发板光盘中了，路径为：开发板光盘 A-基础资料→4、软件→FileZilla_3.60.1_win64-setup.exe，双击安装即可。安装完成以后找到安装目录，找到图标，然后发送图标快捷方式到桌面，完成以后如图 3.2.3 所示：


图 3.2.3 FileZilla 图标

打开 FileZilla 软件，界面如图 3.2.4 所示：

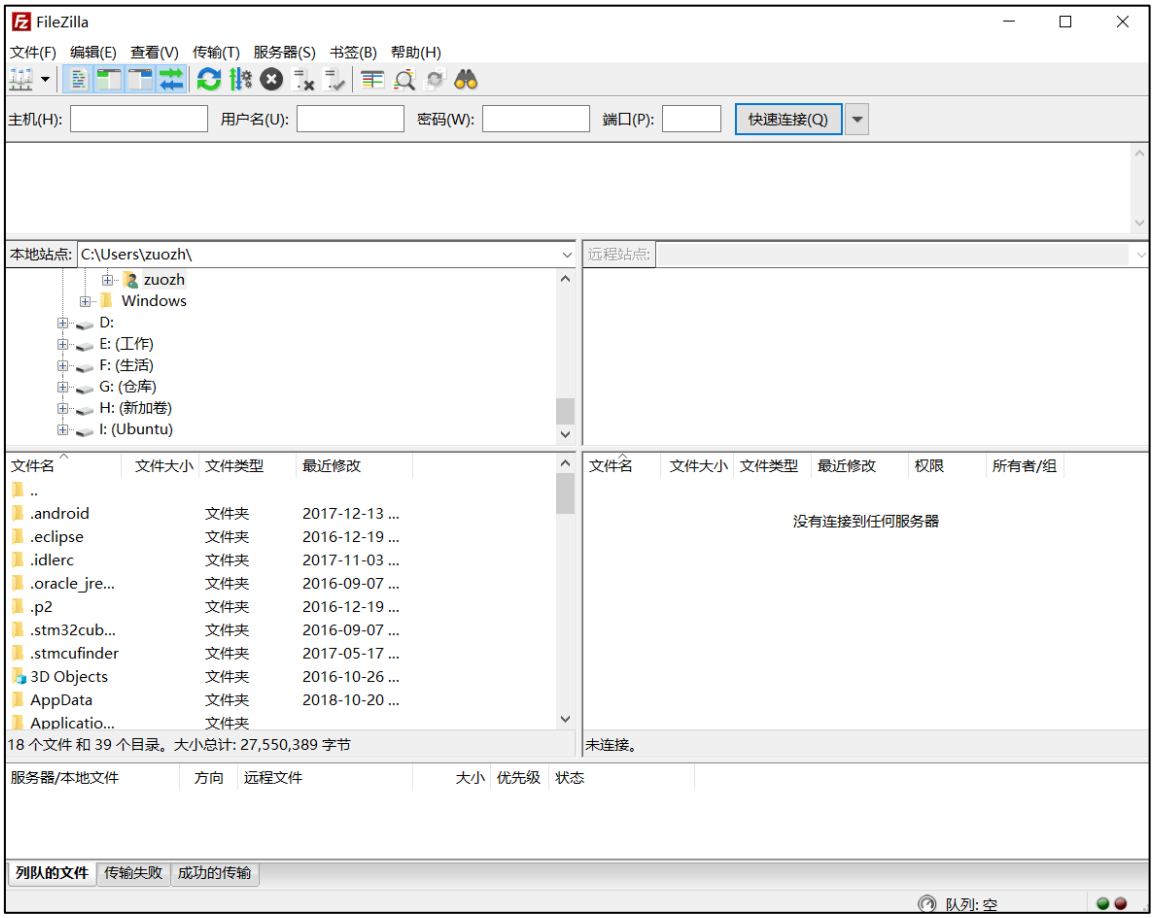



图 3.2.4 FileZilla 软件界面

### 3、FileZilla 软件设置

Ubuntu 作为 FTP 服务器，FileZilla 作为 FTP 客户端，客户端肯定要连接到服务器上，打开站点管理器，点击：文件->站点管理器，打开以后如图 3.2.5 所示：


图 3.2.5 站点管理器

点击图 3.2.5 中的“新站点(N)”按钮来创建站点，新建站点以后就会在“我的站点”下出现新建的这个站点，站点的名称可以自行修改，比如我将新的站点命名为“Ubuntu”如图 3.2.6 所示：

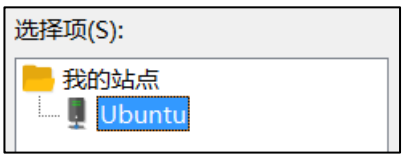



图 3.2.6 新建站点

选中新创建的“Ubuntu”站点，然后对站点的“常规”进行设置，设置如图 3.2.7 所示：


图 3.2.7 站点设置

按照图 3.2.7 中设置好以后，点击“连接”按钮，第一次连接可能会弹出提示是否保存密码的对话框，点击确定即可。连接成功以后如图 3.2.8 所示：

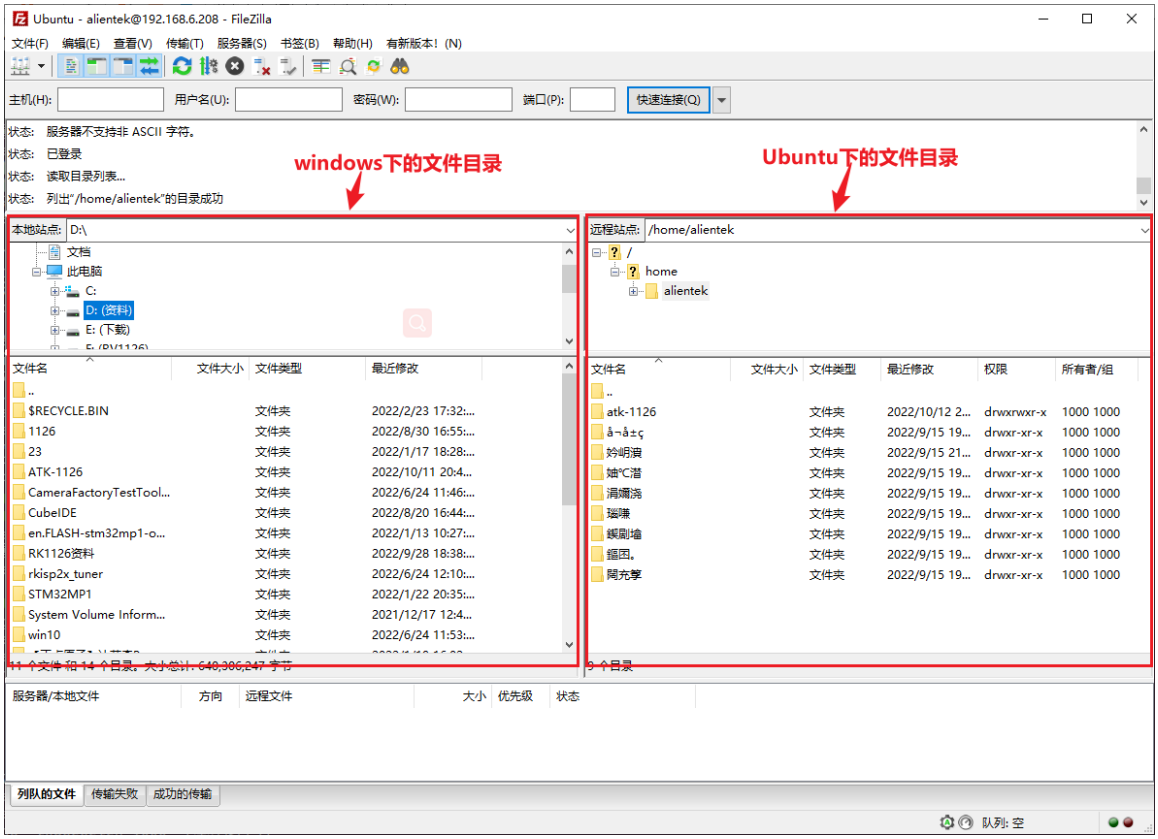



图 3.2.8 连接成功

连接成功以后如图 3.2.8 所示，其中左边就是 Windows 文件目录，右边是 Ubuntu 文件目录，默认进入用户根目录下（比如我电脑的“/home/alientek”）。但是注意观察在图 3.2.8 中 Ubuntu 文件目录下的中文目录都是乱码的，这是因为编码方式没有选对，先断开连接，点击：服务器(S)->断开连接，然后打开站点管理器，选中要设置的站点“Ubuntu”，选择“字符集”，设置如图 3.2.9 所示：

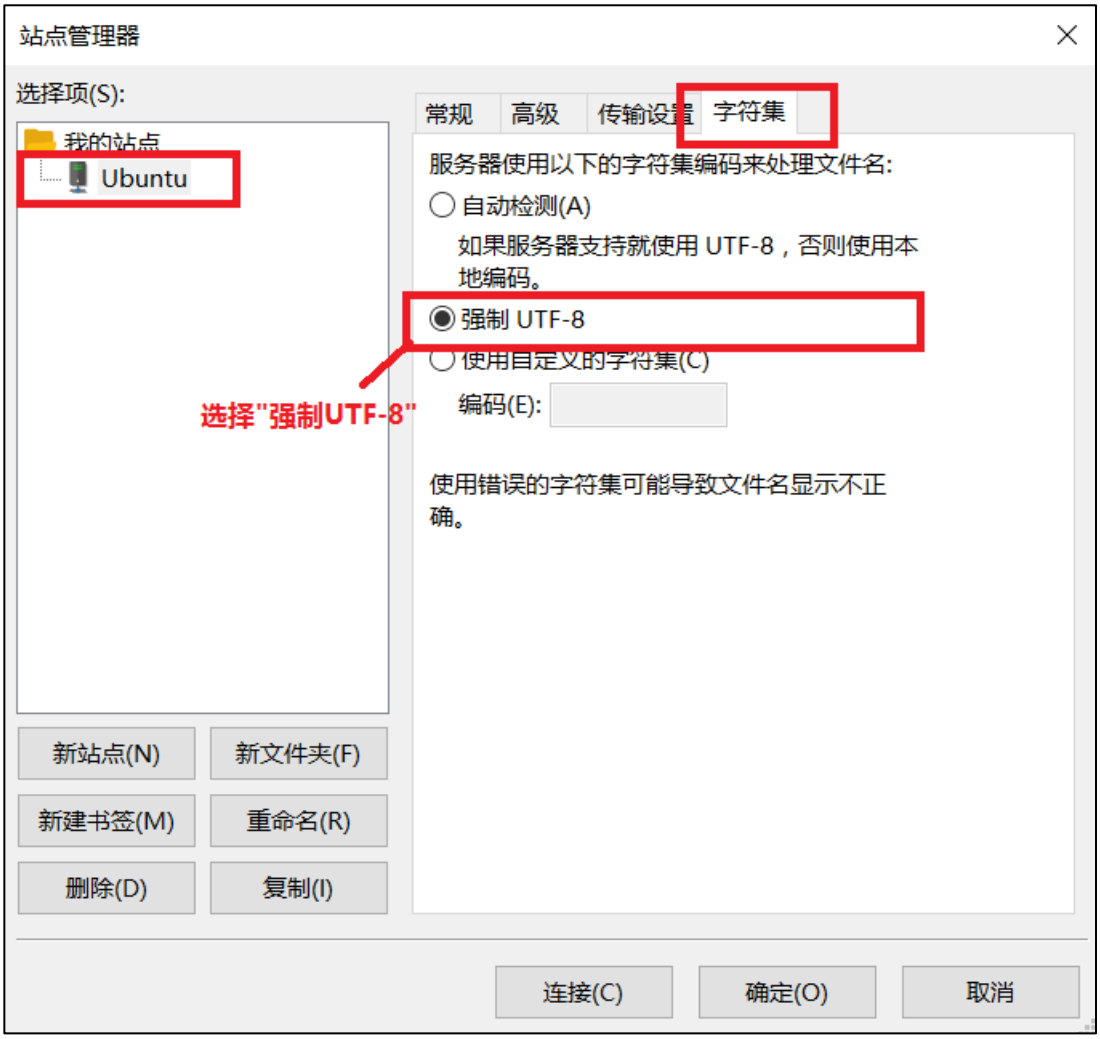



图 3.2.9 设置字符集

按照图 3.2.9 设置好字符集以后重新连接到 FTP 服务器上，重新链接到 FTP 服务器以后 Ubuntu 下的文件目录中文显示就正常了，如图 3.2.10 所示：

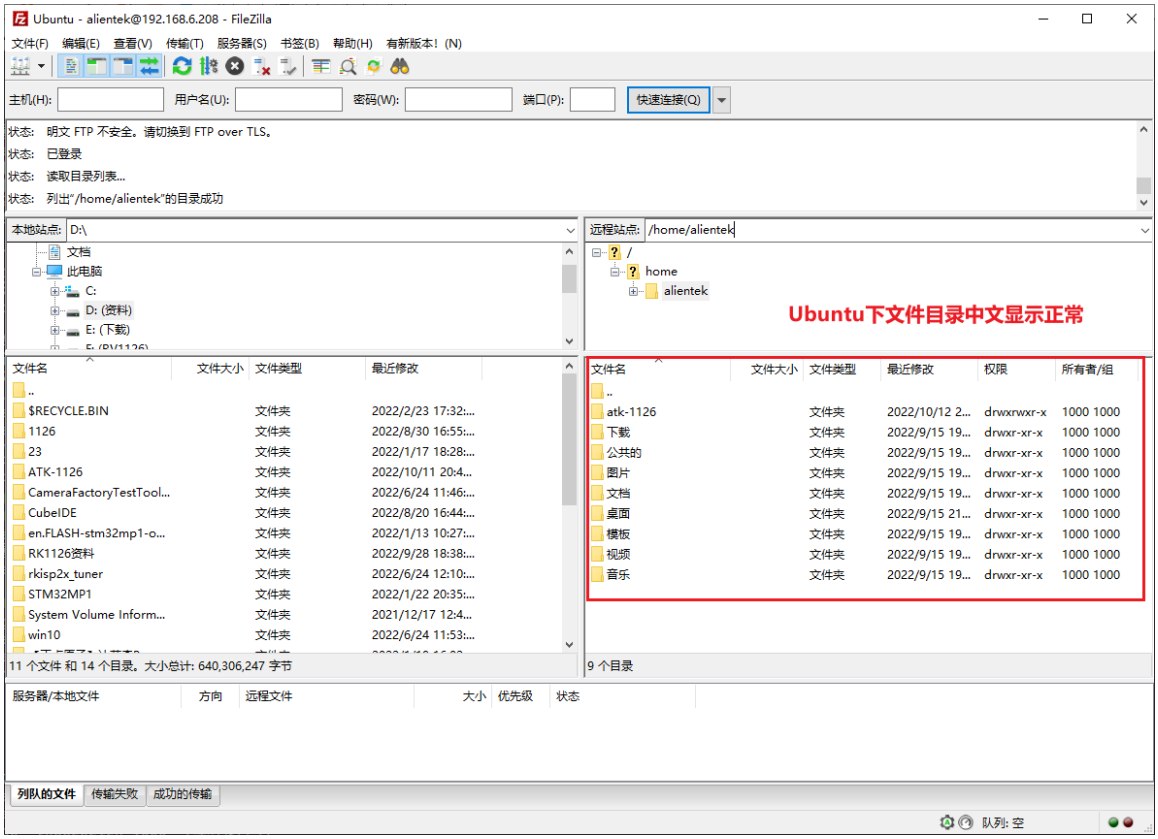



图 3.2.10 Ubuntu 下文件目录中文显示正常

如果要将 Windows 下的文件或文件夹拷贝到 Ubuntu 中，只需要在图 3.2.10 中左侧的 Windows 区域选中要拷贝的文件或者文件夹，然后直接拖到右侧的 Ubuntu 中指定的目录即可。将 Ubuntu 中的文件或者文件夹拷贝到 Windows 中也是直接拖放。

## 3.3 Visual Studio Code 软件的安装和使用

### 3.3.1 Visual Studio Code 的安装

Visual Stuio Code 是一个编辑器，可以用来编写代码，Visual Studio Sode 本教程以后就简称为 VSCode，VSCode 是微软出的一款免费编辑器。VSCode 有 Windows、Linux 和 macOS 三个版本的，是一个跨平台的编辑器。VSCode 下载地址是：https://code.visualstudio.com/Download，下载界面如图 3.3.1 所示：

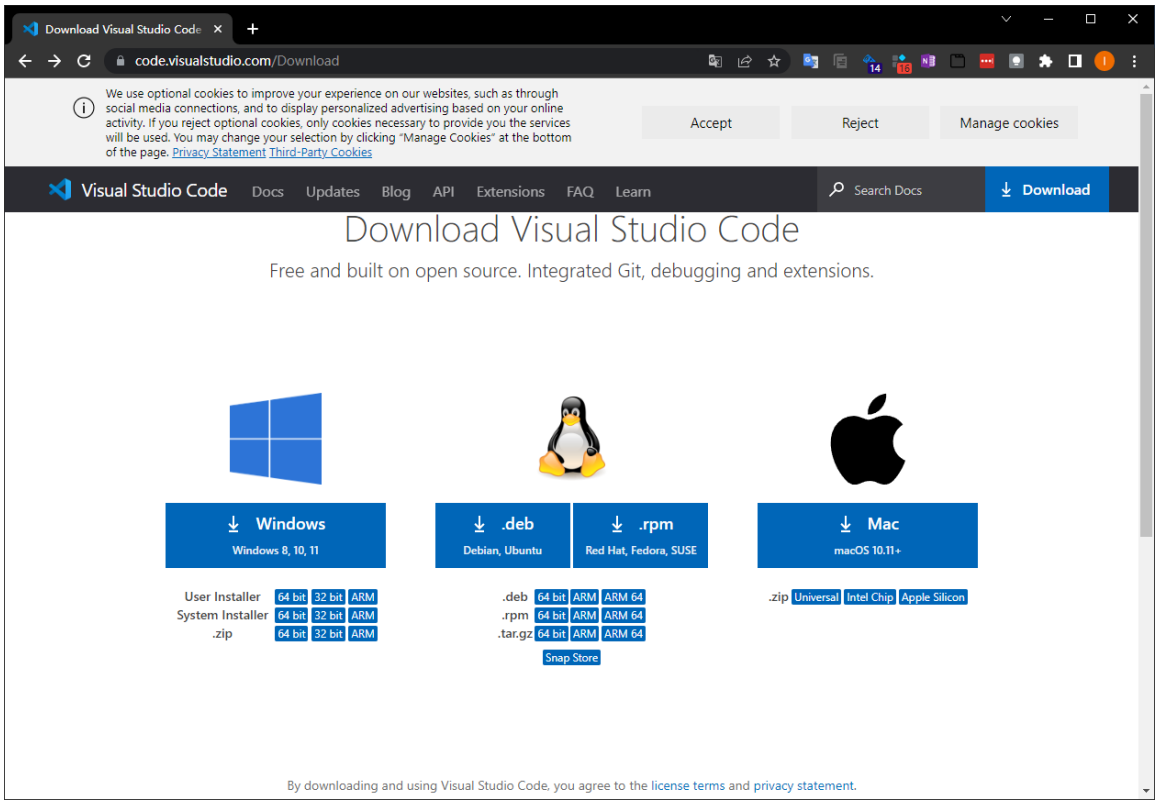



图 3.3.1.1 VSCode 下载界面

在图 3.3.1.1 中下载自己想要的版本，本教程需要 Windows 和 Linux 这两个版本，所以下载这两个即可，我们已经下载好并放入了开发板光盘中，路径为：开发板光盘 A-基础资料→4、软件→Visual Studio Code。

**1、Windows 版本安装**

Windows 版本的安装和容易，和其他 Windows 一样，双击.exe 安装包，然后一路“下一步”即可，安装完成以后在桌面上就会有 VSCode 的图标，如图 3.3.1.2 所示：


图 3.3.1.2 VSCode 图标

双击图 3.3.1.2 打开 VSCode，默认界面如图 3.3.1.3 所示：

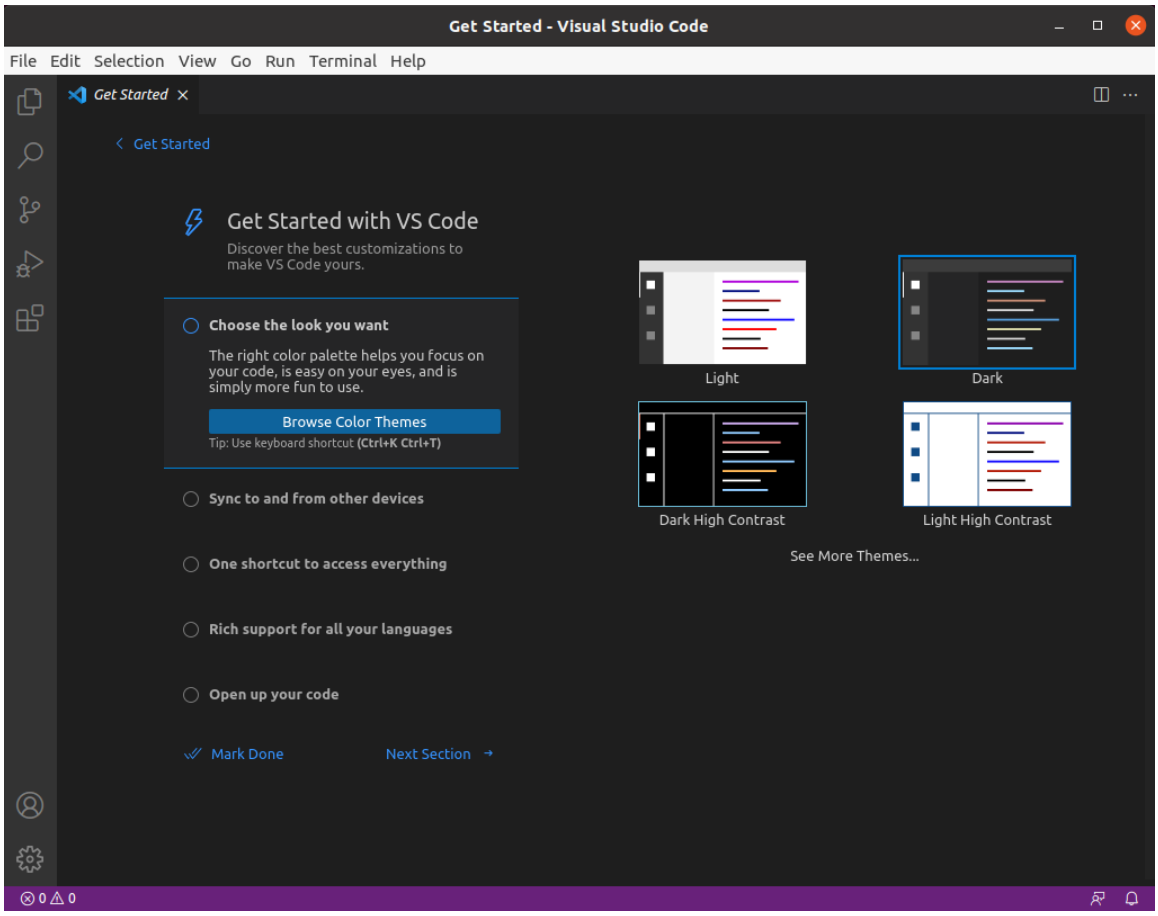



图 3.3.1.3 VSCode 默认界面

**2、Linux 版本安装**

我们有时候也需要在 Ubuntu 下阅读代码，所以还需要在 Ubuntu 下安装 VSCode。Linux 下的 VSCode 安装包我们也放到了开发板光盘中， 将开发板光盘中的.deb 软件包拷贝到 Ubuntu 系统中，然后使用如下命令安装：

```
sudo dpkg -i code_1.72.1-1665423861_amd64.deb
```

等待安装完成，如图 3.3.1.4 所示：

```
alientek@alientek-virtual-machine:~$ sudo dpkg -i code_1.72.1-1665423861_amd64.deb
[sudo] alientek 的密码:
正在选中未选择的软件包 code。
(正在读取数据库 ... 系统当前共安装有 247655 个文件和目录。)
准备解压 code_1.72.1-1665423861_amd64.deb  ...
正在解压 code (1.72.1-1665423861) ...
正在设置 code (1.72.1-1665423861) ...
正在处理用于 gnome-menus (3.36.0-1ubuntu1) 的触发器 ...
正在处理用于 desktop-file-utils (0.24-1ubuntu3) 的触发器 ...
正在处理用于 mime-support (3.64ubuntu1) 的触发器 ...
正在处理用于 shared-mime-info (1.15-1) 的触发器 ...
alientek@alientek-virtual-machine:~$
```

图 3.3.1.4 VSCode 安装过程

安装完成以后在终端下运行“code”命令即可打开，如下所示：

```
code
```

结果如下图所示：

```
alientek@alientek-virtual-machine:~$
alientek@alientek-virtual-machine:~$ code
alientek@alientek-virtual-machine:~$
```

运行成功后会弹出如下所示:

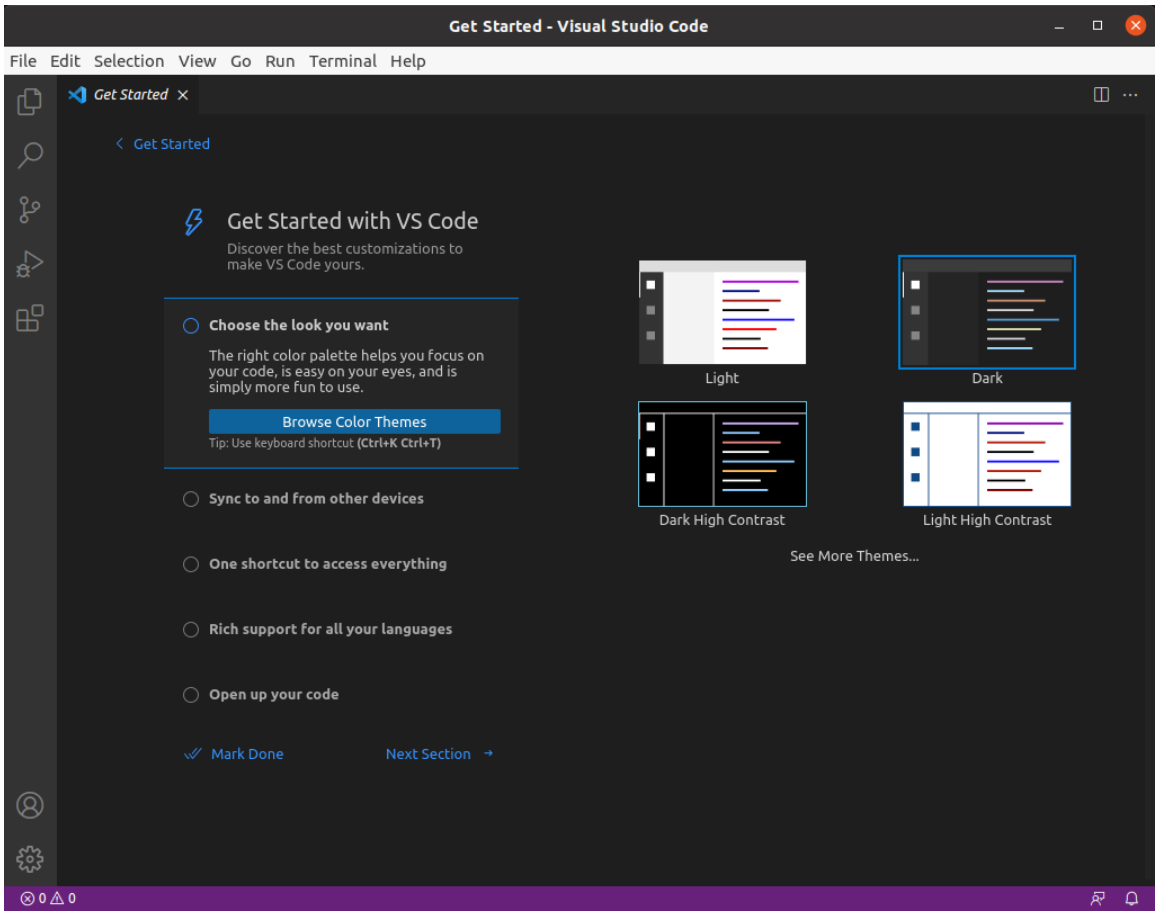



图 4.5.1.9 Linux 下的 VSCode

可以看出 Linux 下的 VSCode 和 Windows 下的基本是一样的，所以使用方法也是一样的。

### 3.3.2 Visual Studio Code 插件的安装

VSCode 支持多种语言，比如 C/C++、Python、C#等等，本教程我们主要用来编写 C/C++程序的，所以需要安装 C/C++的扩展包，扩展包安装很简单，如图 3.3.2.1 所示：

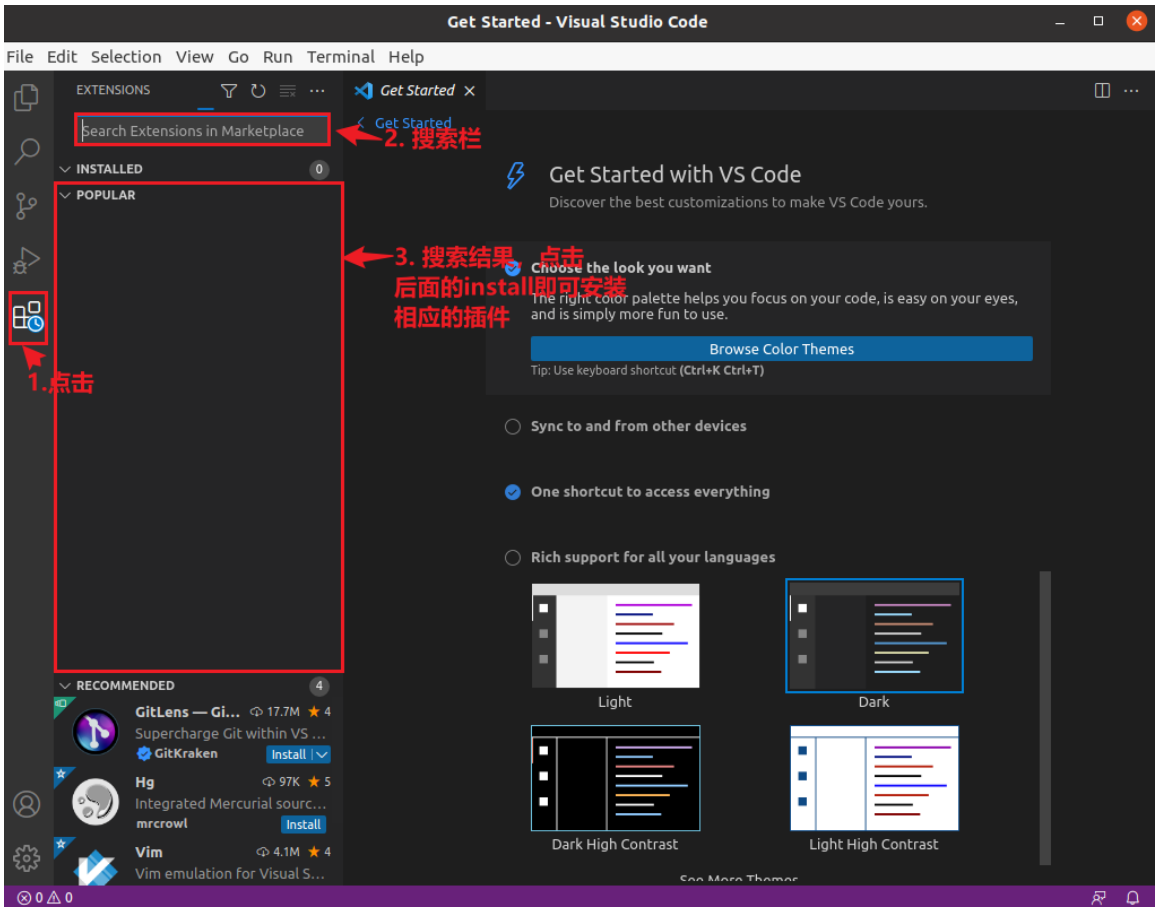



图 3.3.2.1 VSCode 插件安装

我们需要按照的插件有下面几个：

1)、C/C++，这个肯定是必须的。

2)、C/C++ Snippets，即 C/C++重用代码块。

3)、C/C++ Advanced Lint,即 C/C++静态检测 。

4)、Code Runner，即代码运行。

5)、Include AutoComplete，即自动头文件包含。

6)、Rainbow Brackets，彩虹花括号，有助于阅读代码。

7)、One Dark Pro，VSCode 的主题。

8)、GBKtoUTF8，将 GBK 转换为 UTF8。

9)、ARM，即支持 ARM 汇编语法高亮显示。

10)、Chinese(Simplified)，即中文环境。

11)、vscode-icons，VSCode 图标插件，主要是资源管理器下各个文件夹的图标。

12)、compareit，比较插件，可以用于比较两个文件的差异。

13)、DeviceTree，设备树语法插件。

14)、TabNine，一款 AI 自动补全插件，强烈推荐，谁用谁知道！

15)、Remote-SSH，可以远程连接到别的 vscode 上的软件。

安装完成以后重新打开 VSCode，如果要查看已经安装好的插件，可以按照图 3.3.2.2 所示方法查看：

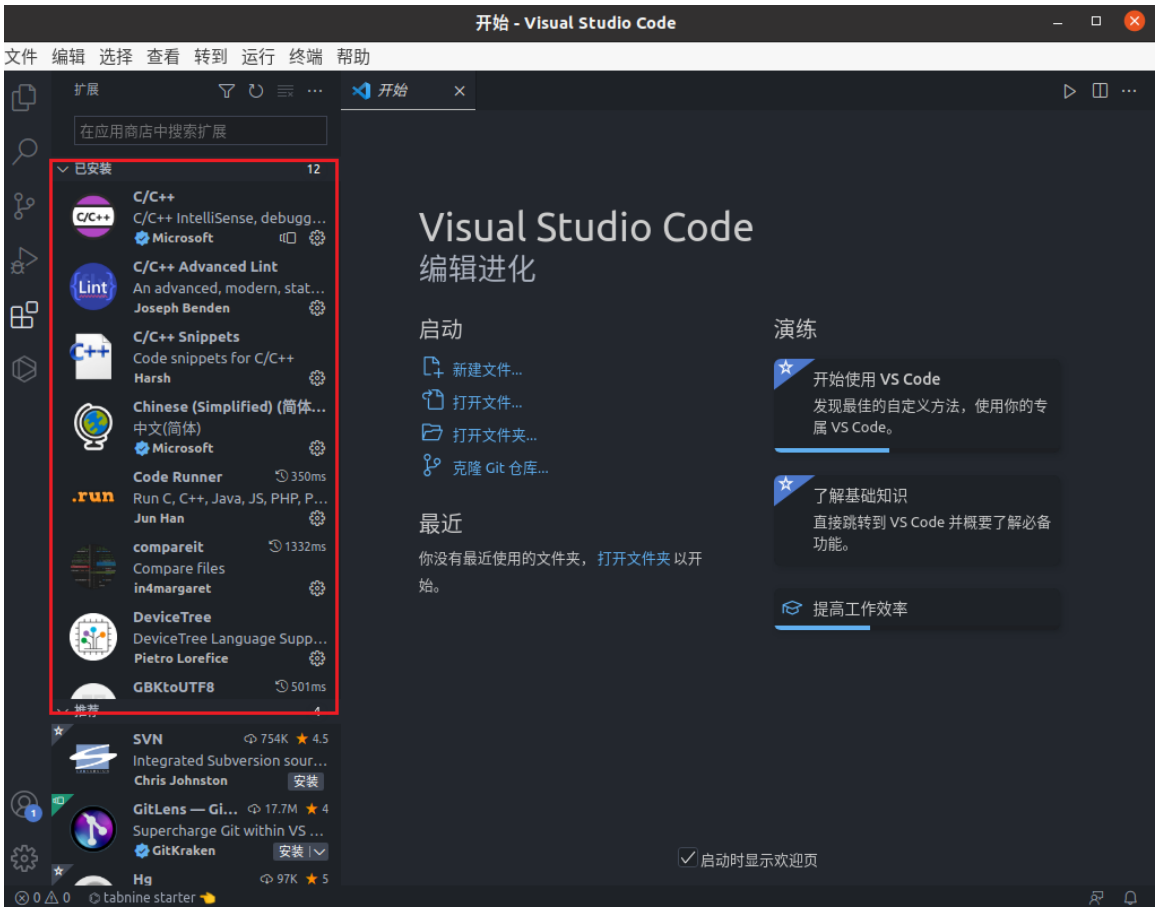



图 3.3.2.2 显示已安装的插件

### 3.3.3 vscode 远程 Ubuntu 系统下的 vscode

ATK-DLRV1126 的开发环境必须在 linux 系统下进行开发。在开发的时候需要切换到 Ubuntu 系统，工作写文档的时候就切换回 Windows 系统，这样是很麻烦的，我们可以使用 vscode 的远程功能插件“Remote-SSH”，此插件可以进行远程开发。使用此插件前有两个前提：本地端、远程端需要安装 vscode、本地和远程网络可以相互 ping 通(我们安装的是虚拟机，使用桥接网络即可实现)。首先打开 Windows 下的 vscode。点击左下角绿色图标打开远程窗口，如下图所示：

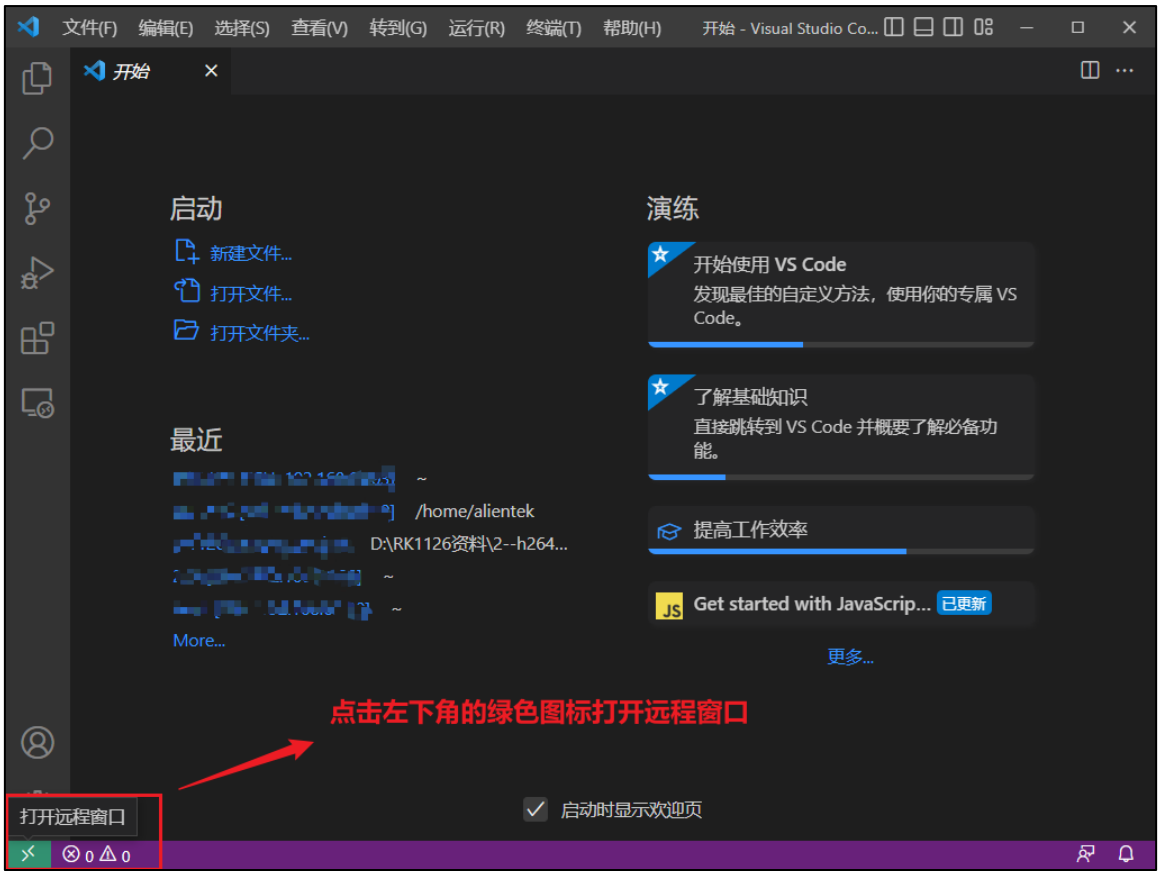



图 3.3.3.1 打开远程窗口

打开远程窗口，就会出现如下图所示：

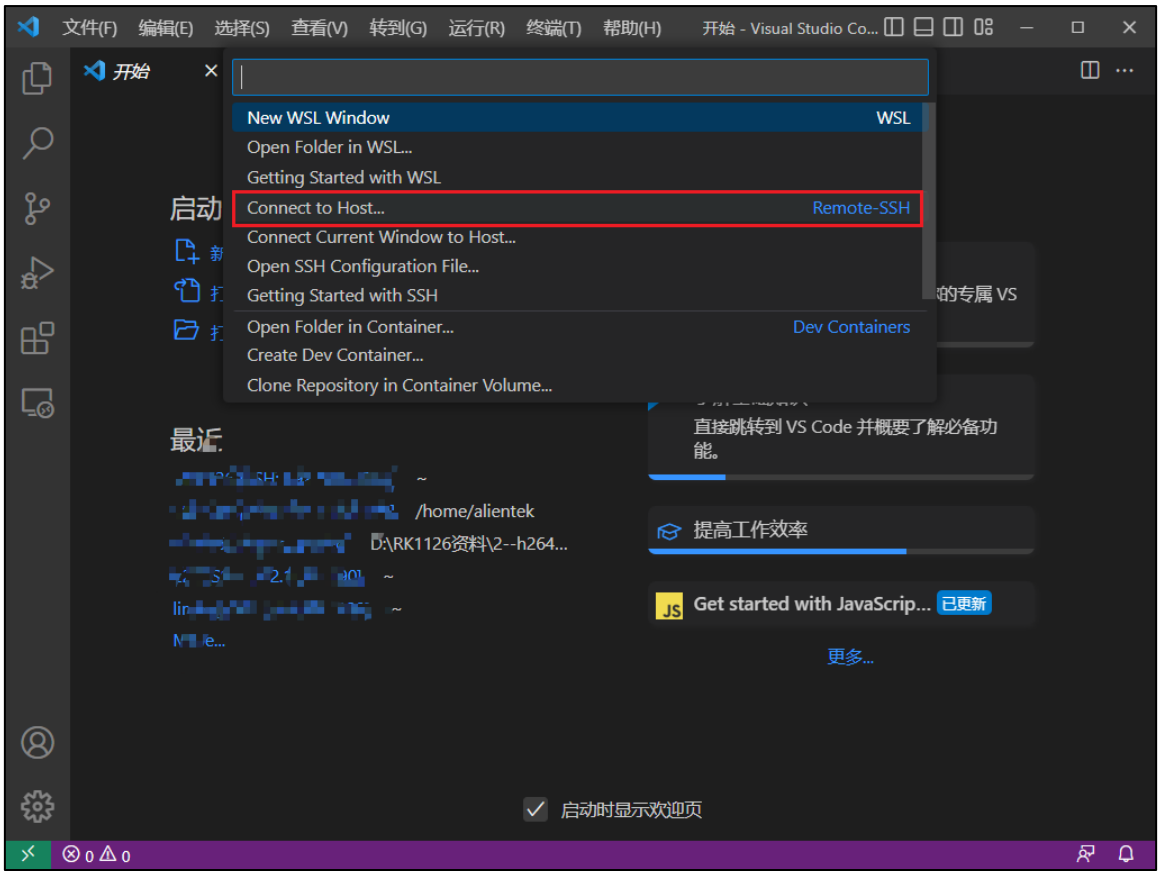



图 3.3.3.2 远程功能的选择

选择图 3.3.3.2 中的红色框“Connect to Host”功能，如下图所示：

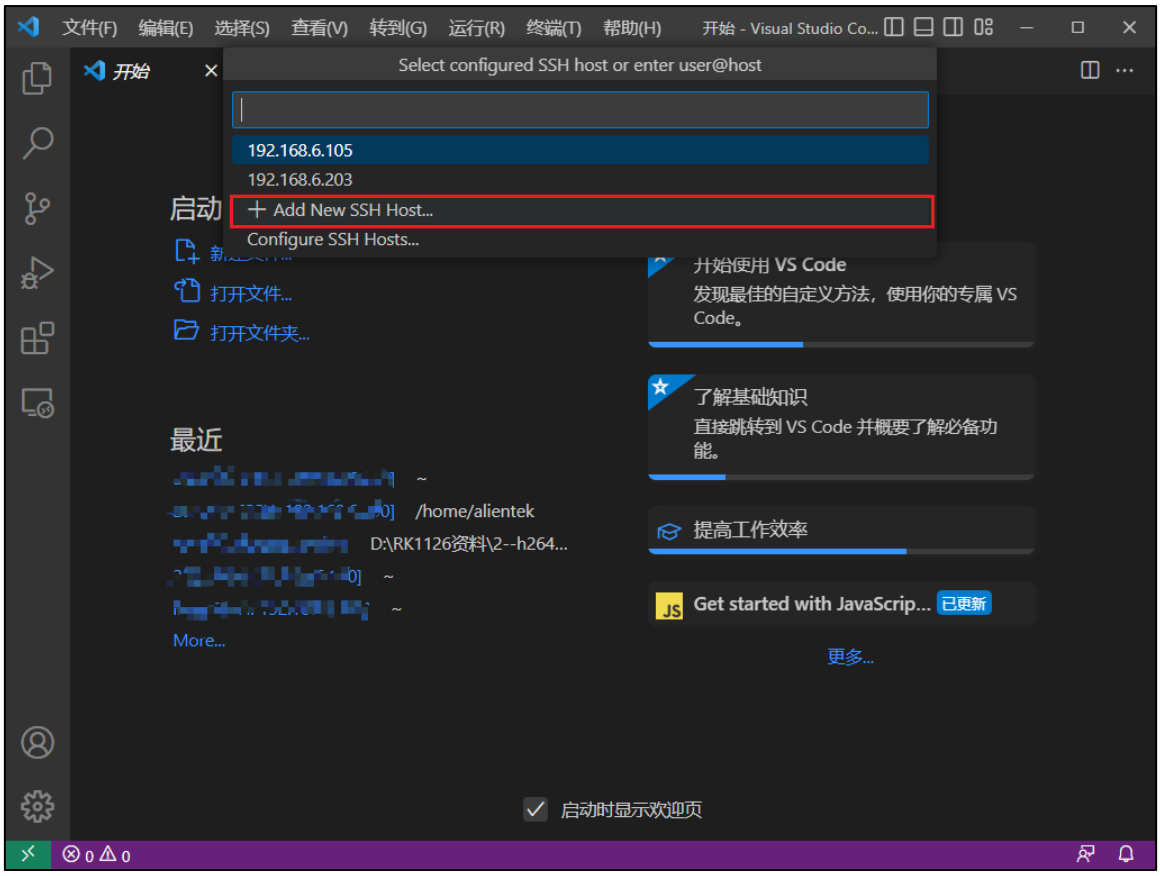



图 3.3.3.3 添加 SSH 的配置

选择图 3.3.3.3 中的红色框“Add New SSH Host”功能。如下图所示：


图 3.3.3.4 添加新的远程连接

根据图中的红色框提示信息输入远程电脑的用户名和 IP 地址，这边笔者的用户名和 IP 地址分别为：alientek 和 192.168.6.208。输入如下命令即可连接：

```
ssh alientek@192.168.6.208 -A
```

输入连接命令后，按回车键，如下图所示：

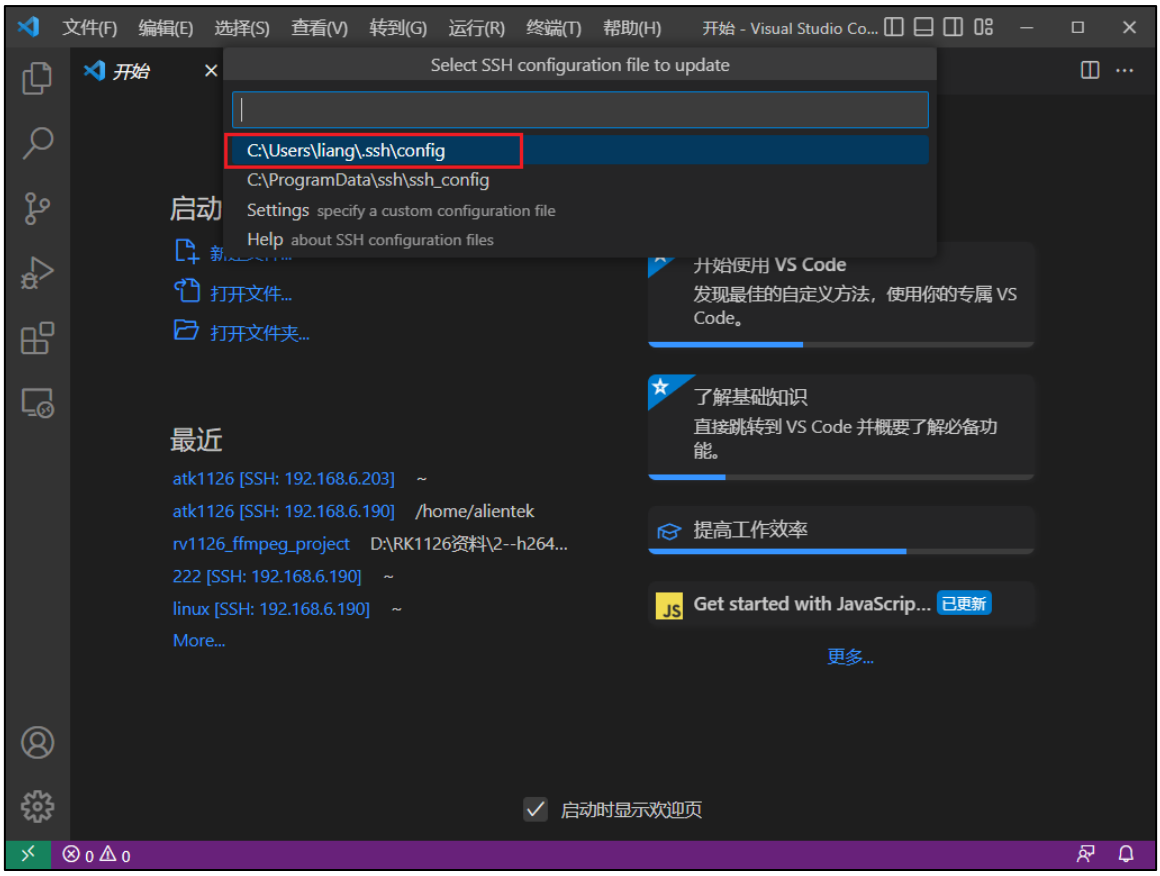



图 3.3.3.5 选择保存配置文件

图 3.3.3.5 中，主要是要保存刚刚输入的配置到那个文件下，通常选择红色框“C:\Users\liang\.ssh\config”的路径文件。配置完成后，就会弹出如下图所示的小框：

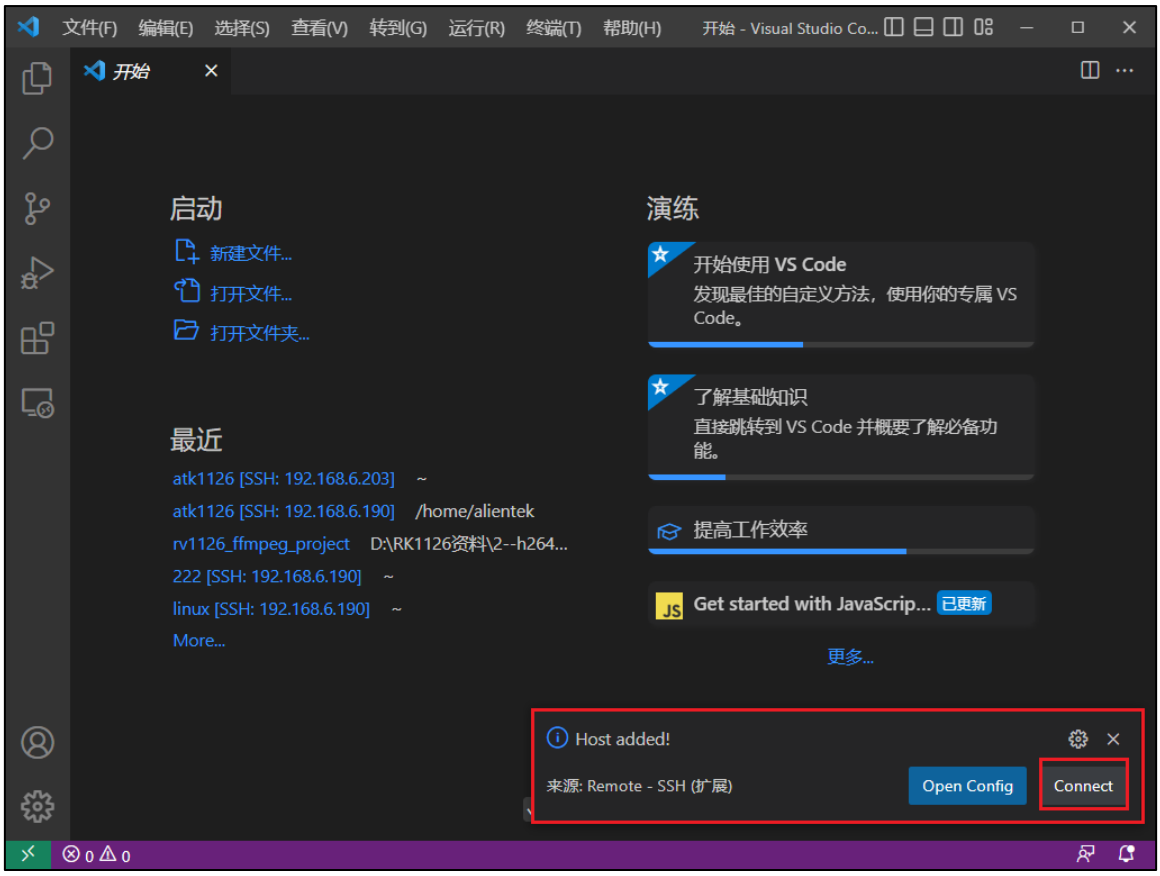



图 3.3.3.6 连接远程端的 vscode

点击图 3.3.3..6 中的右下角“Connect”，即可进入连接状态，如下图所示：

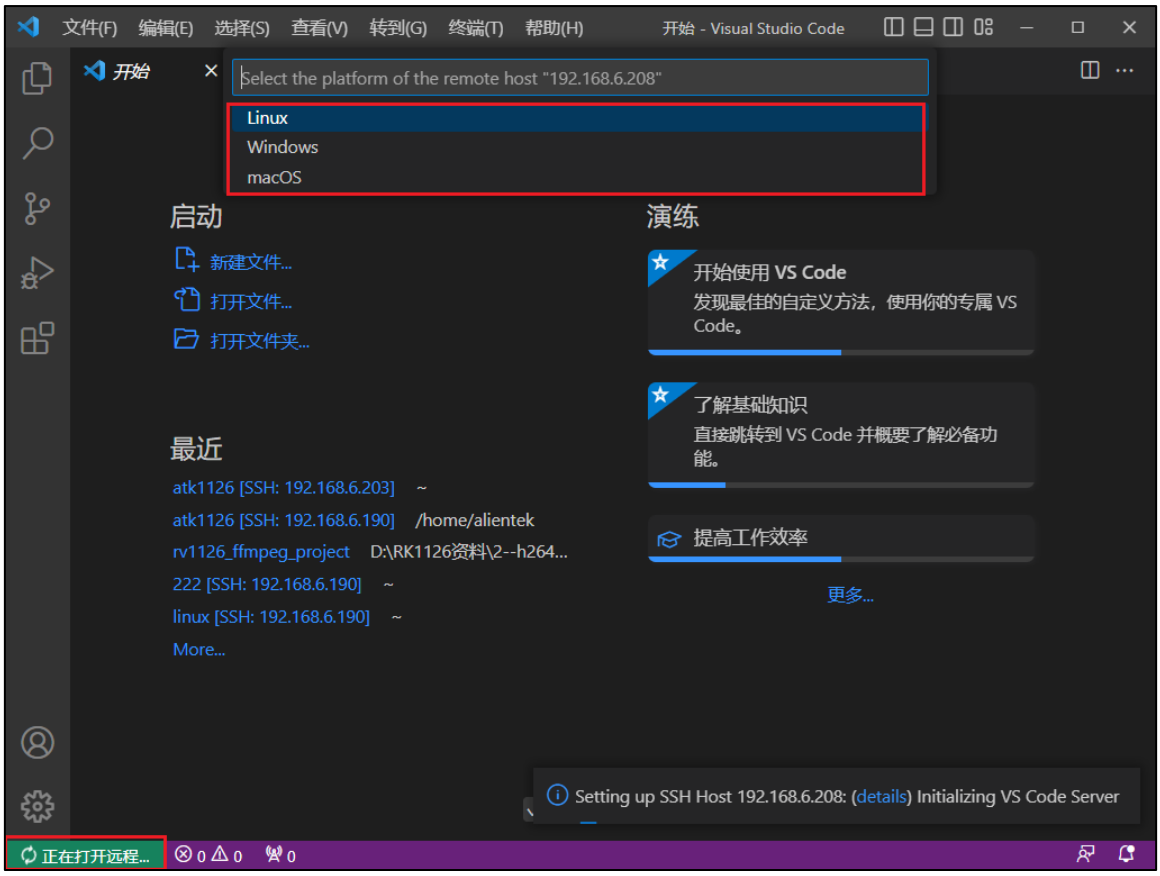



图 3.3.3.7 选择远程电脑的系统

图 3.3.3.7 中，开始进行远程连接配置，首先要选择远程电脑的系统是什么，这边我们连接的是 Ubuntu，所以选择“Linux”。左下角开始显示“正在打开远程”，选择完，就会出现如下图所示：

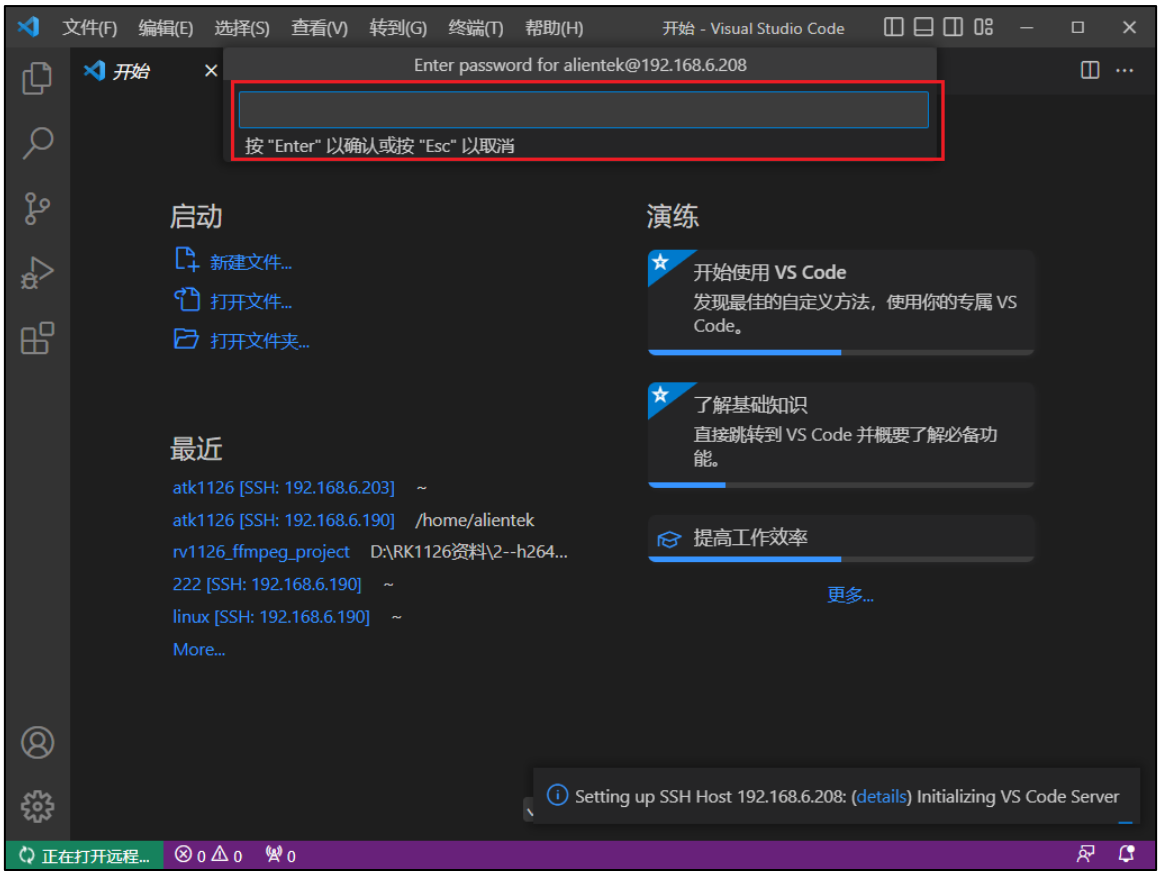



图 3.3.3.8 输入远程登录密码

按照图 3.3.3.8 中，输入远程端的电脑密码，按回车键，就能弹出新的 vscode 窗口，显示连接成功，如下图所示：

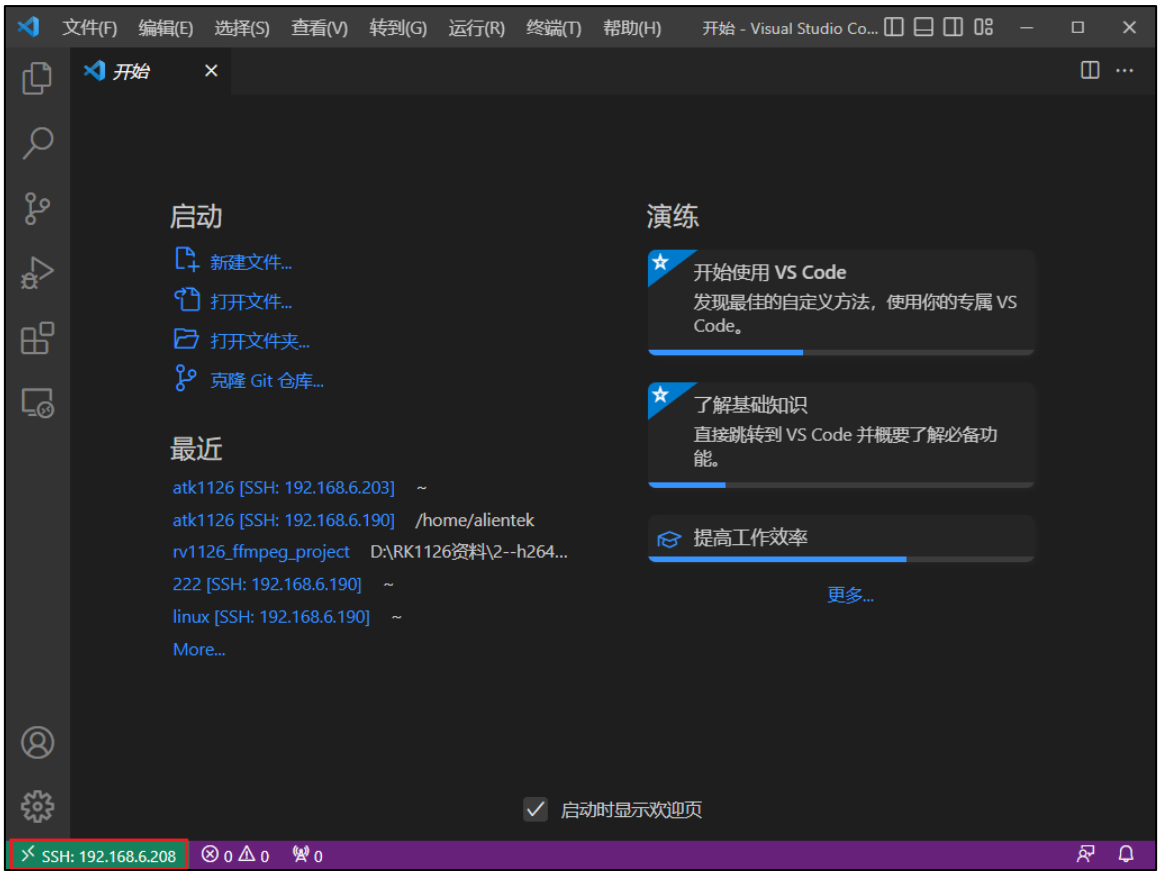



图 3.3.3.9 远程到 192.168.6.208 Ubuntu 系统下的 vscode

图 3.3.3.9 中的左下角里面，已经远程 Ubuntu 系统了。

### 3.3.4 vscode 的使用

本小节主要是教大家如何通过 vscode 远程的方式，打开 ATK-DLRV1126 开发板的源码阅读和编译。SDK 的源码目录在开发板光盘 A-基础资料→01、程序源码→01、正点原子 SDK 源码→atk-rv1126_linux_release_v1.1_2022127.tar.bz2，拷贝此文件到 Ubuntu，解压到 Ubuntu 下。根据 3.3.3 小节使用 vscode 远程连接 Ubuntu 系统的 vscode，点击文件，进入下所示：

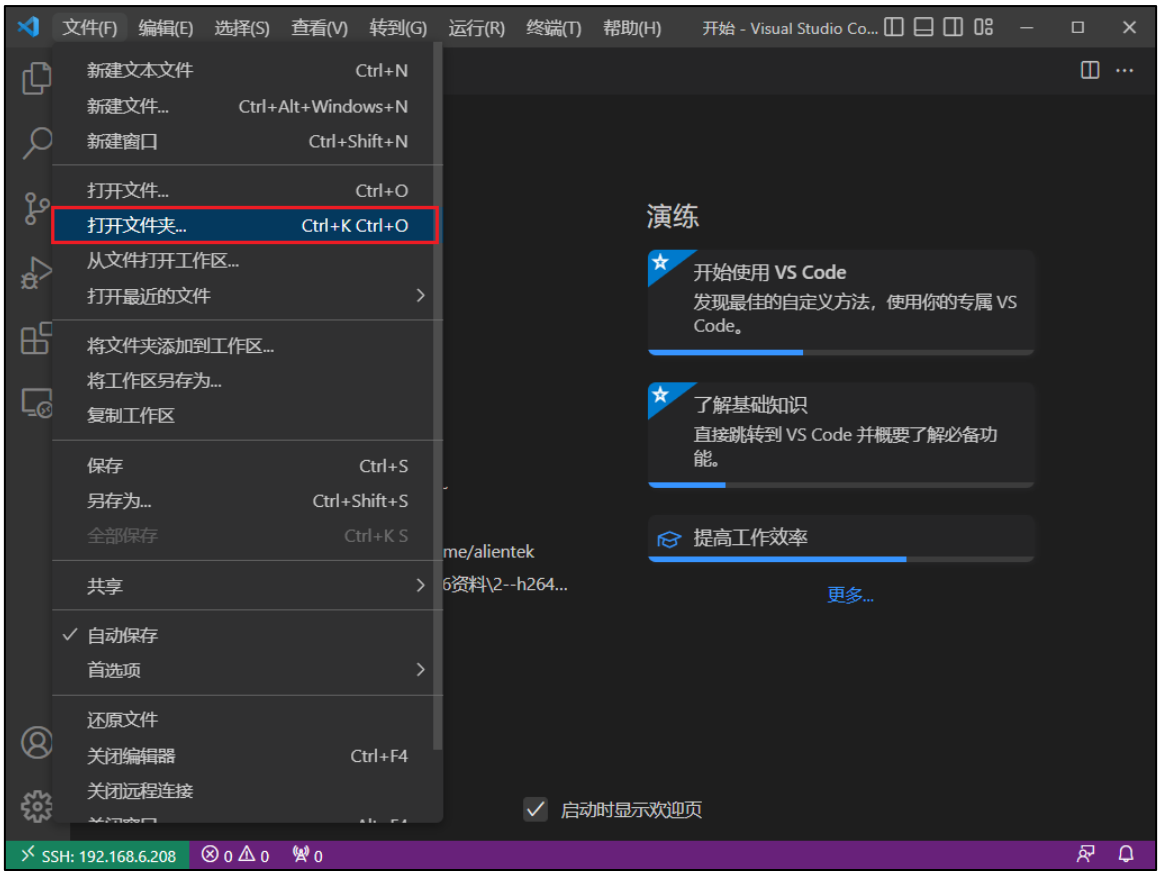



图 3.3.4.1 打开文件夹

点击图 3.3.4.1 中的“打开文件夹”如下图所示：

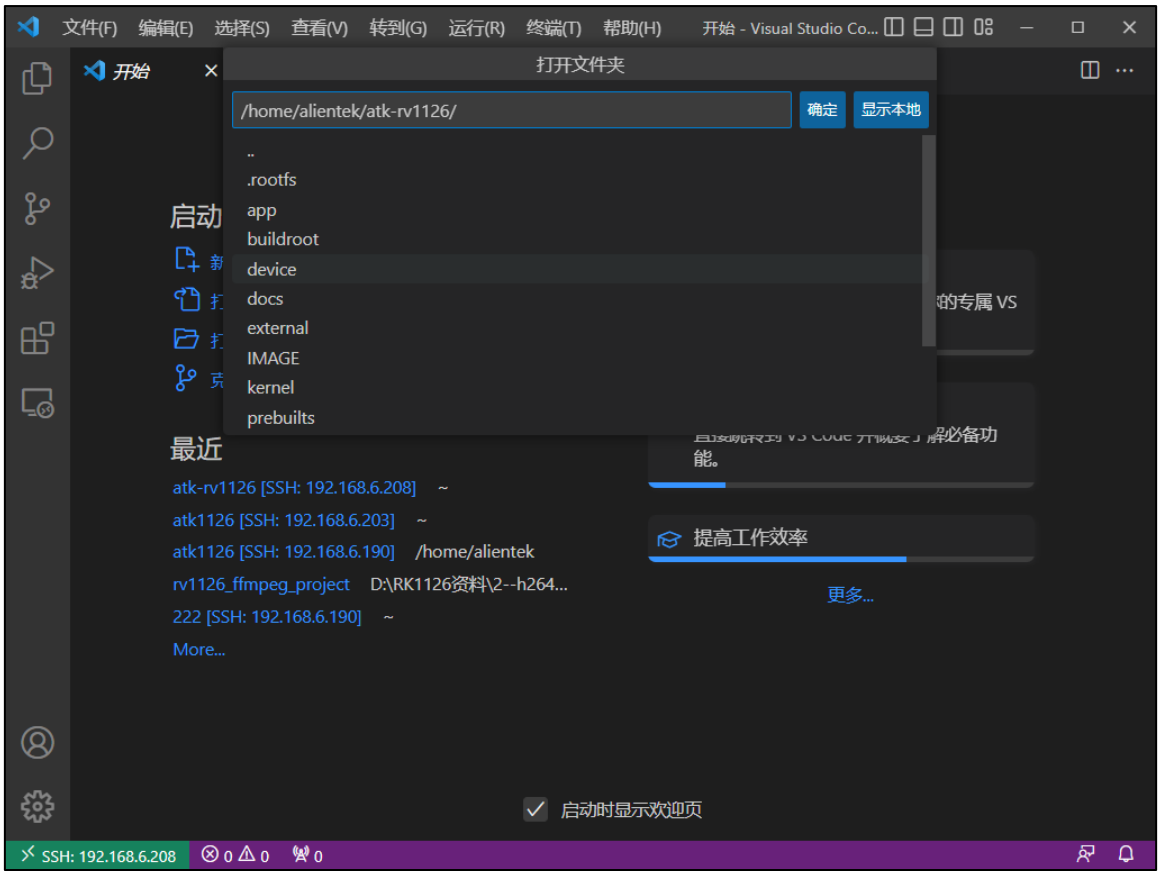



图 3.3.4.2 选择 SDK 包的文件夹

笔者把 SDK 包的源码目录解压到“/home/alientek/atk-rv1126”所以我们在输入框中输入此路径，点击“确认”，然后就要我们输入密码，输入密码按回车键，进入如下图所示：

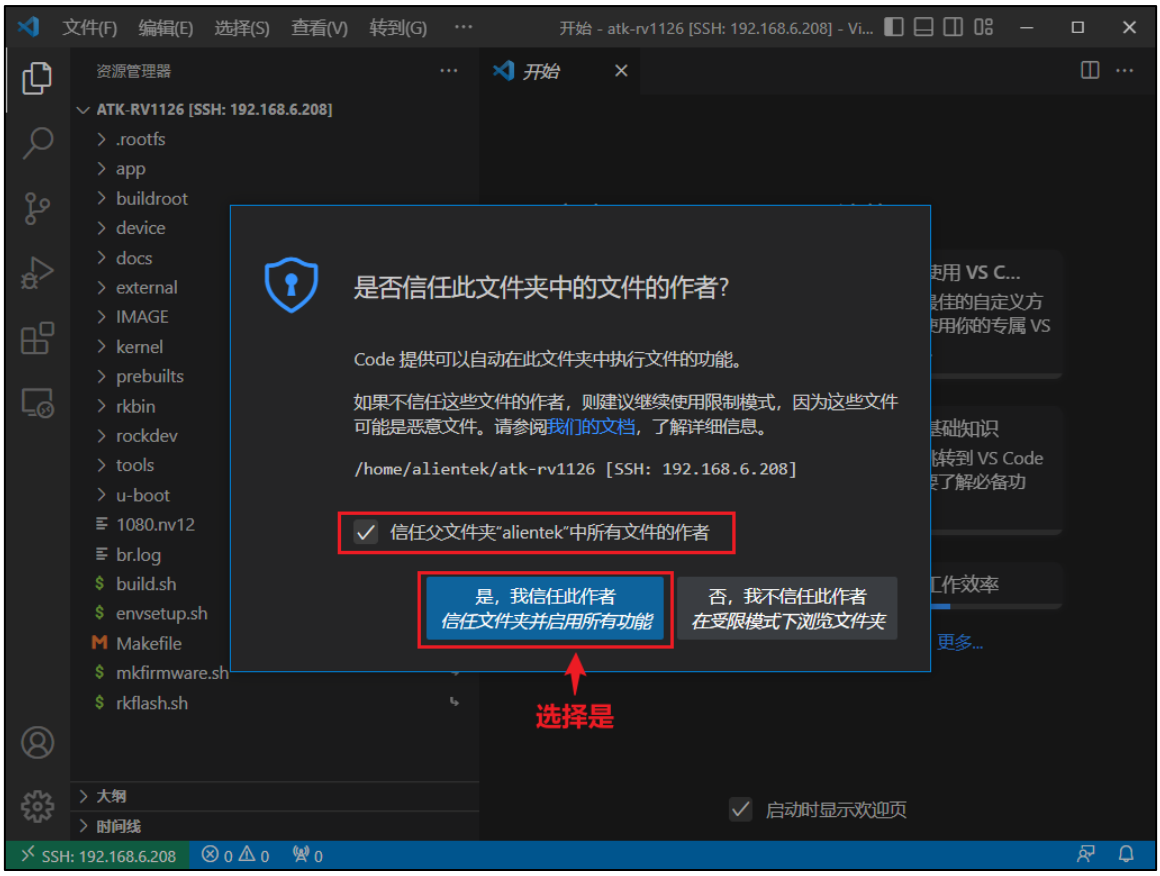



图 3.3.4.3 信任文件夹

图 3.3.4.3 中我们选择是信任文件夹，就会有如下图所示：

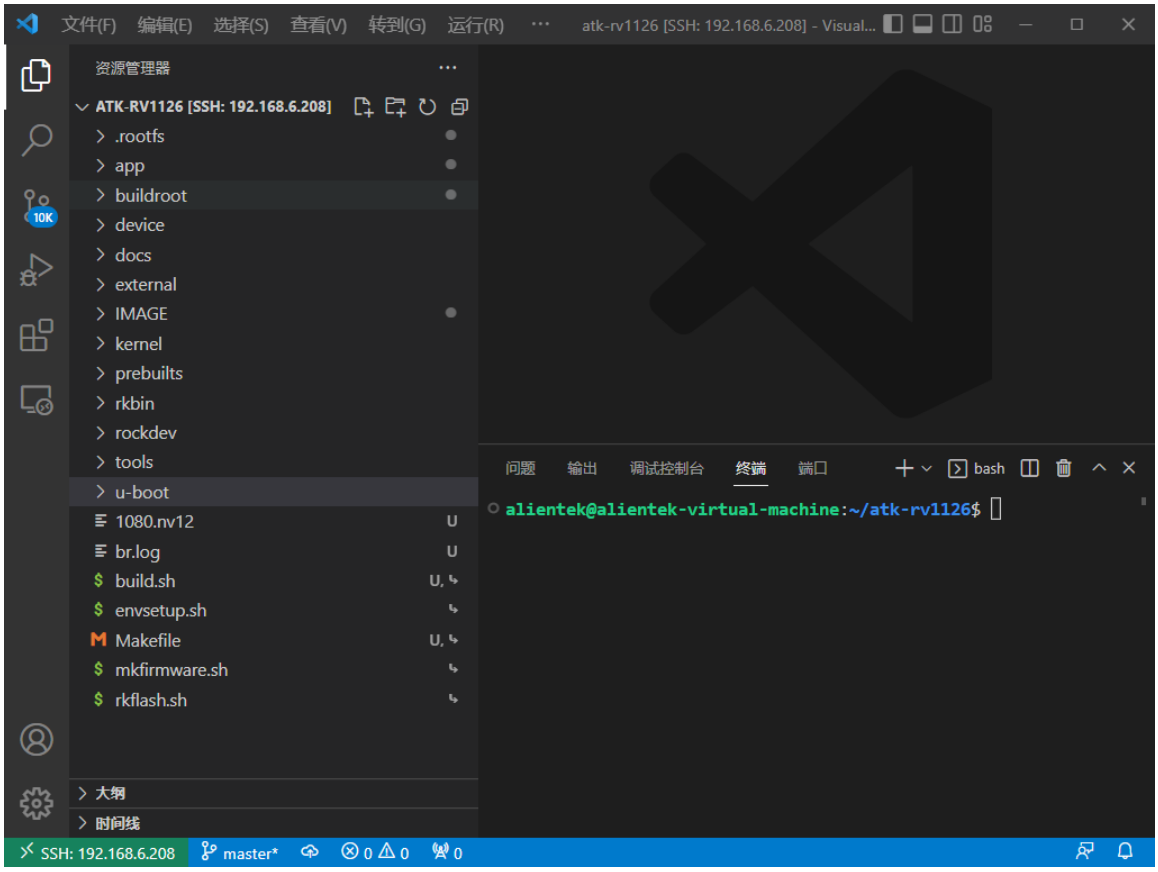



图 3.3.4.4 远程浏览 Ubuntu 系统下的 SDK 代码

此时我们就可以使用 Windows 系统下的 vscode 阅读 Ubuntu 下的代码了，不用切换系统，还能打开终端进行 SDK 包的源码编译。

## 3.4 CH340 串口驱动安装

我们一般在 Windwos 下通过串口来调试程序，或者使用串口作为终端，ATK1126 开发板使用 CH340 这个芯片实现了 USB 转串口功能，CH340 是一枚江苏沁恒生产的国产芯片，稳定性还是很不错的，这里我们要多多支持国产嘛。

先通过 USB 线将开发板的串口和电脑连接起来起来，连接方式如图 3.4.1：

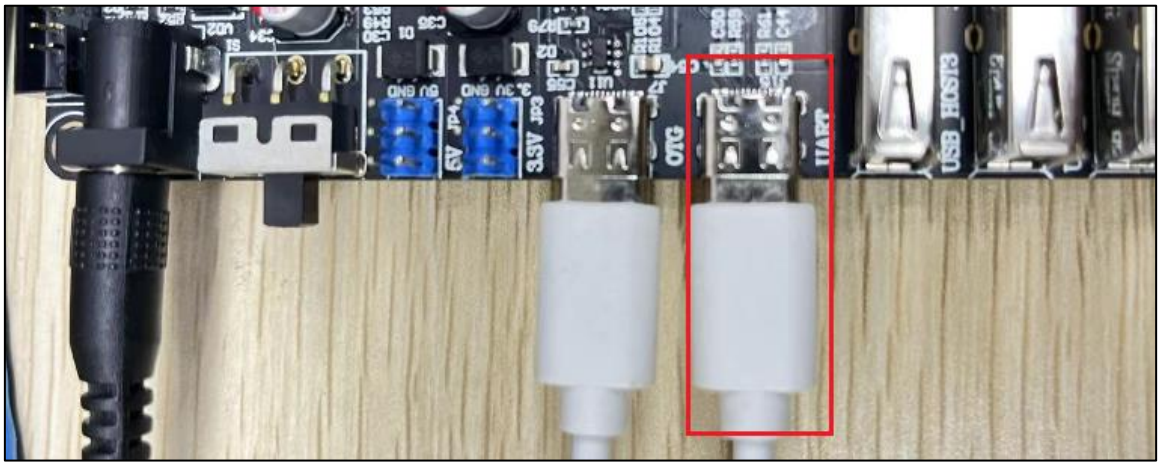


图 3.4.1 开发板串口连接方式

CH340 是需要安装驱动的，驱动我们已经放到了开发板光盘中，路径：开发板光盘 A-基础资料→4、软件→CH340 驱动(USB 串口驱动)_XP_WIN7 共用→SETUP.EXE，，双击 SETUP.EXE，打开如图 4.5.2 所示安装界面：


图 3.4.2 CH340 驱动安装

点击图 3.4.2 中的“安装”按钮开始安装驱动，等待驱动安装完成，驱动安装完成以后会有如图 3.4.3 所示提示：

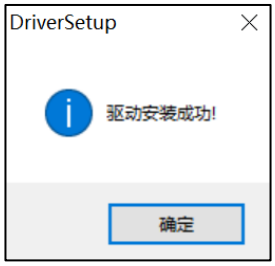



图 3.4.3 驱动安装成功

点击图 3.4.3 中的“确定”按钮退出安装，重新插拔一下串口线。打开设备管理器，打开方式是在 Windows 上的“此电脑”图标上点击鼠标右键，选择“管理”，如图 3.4.4

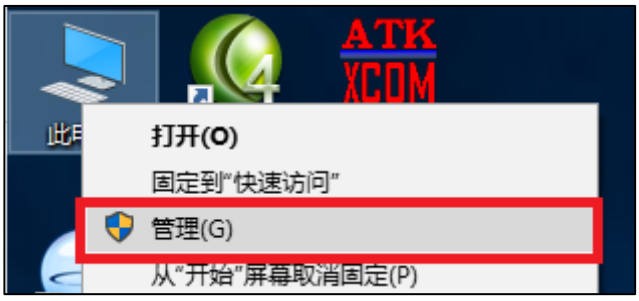



图 3.4.4 打开管理窗口

打开以后的计算机管理器如图 3.4.5 所示：

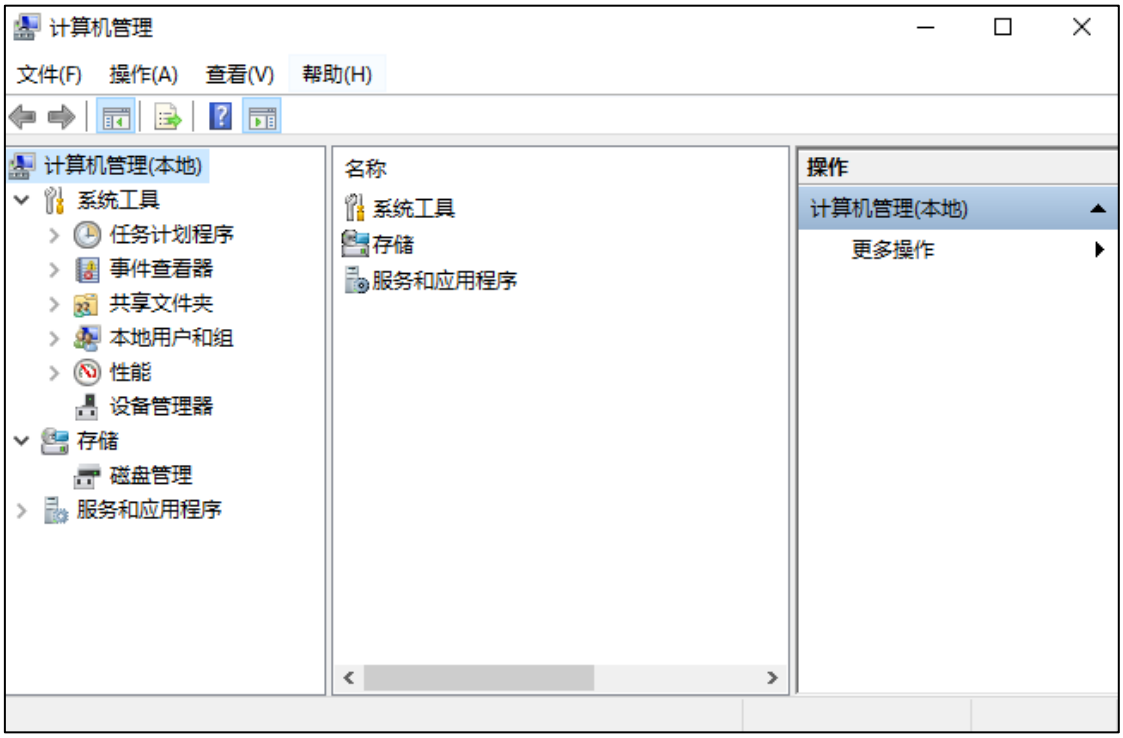



图 3.4.5 计算机管理器

在图 3.4.5 中，点击左侧“计算机管理(本地)”中的“设备管理器”，在右侧选中“端口(COM 和 LPT)”，如图 3.4.6 所示：

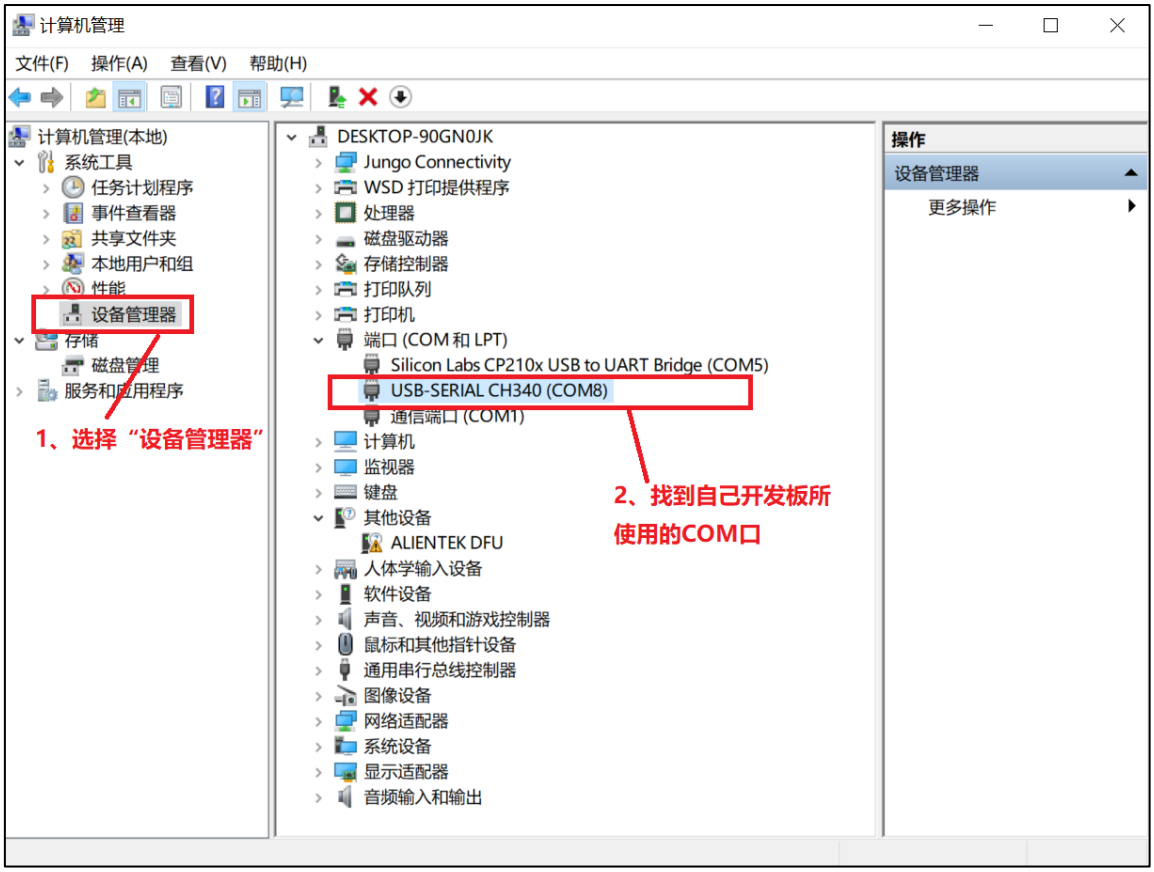



图 3.4.6 设备管理器

如果在图 3.4.6 中找到了有“USB-SERIAL CH340”字样的端口设备就说明 CH340 驱动成功了，一定要用 USB 线将开发板的串口和电脑连接起来！！！！

## 3.5 MobaXterm 软件安装和使用

### 3.5.1 MobaXterm 软件安装

MobaXterm 是一款终端软件，功能强大而且免费(也有收费版)！我试用了一下，用起来非常舒服！在这里推荐大家使用此软件作为终端调试软件，MobaXterm 软件在其官网下载即可，地址为 https://mobaxterm.mobatek.net/，如图 3.5.1.1 所示：

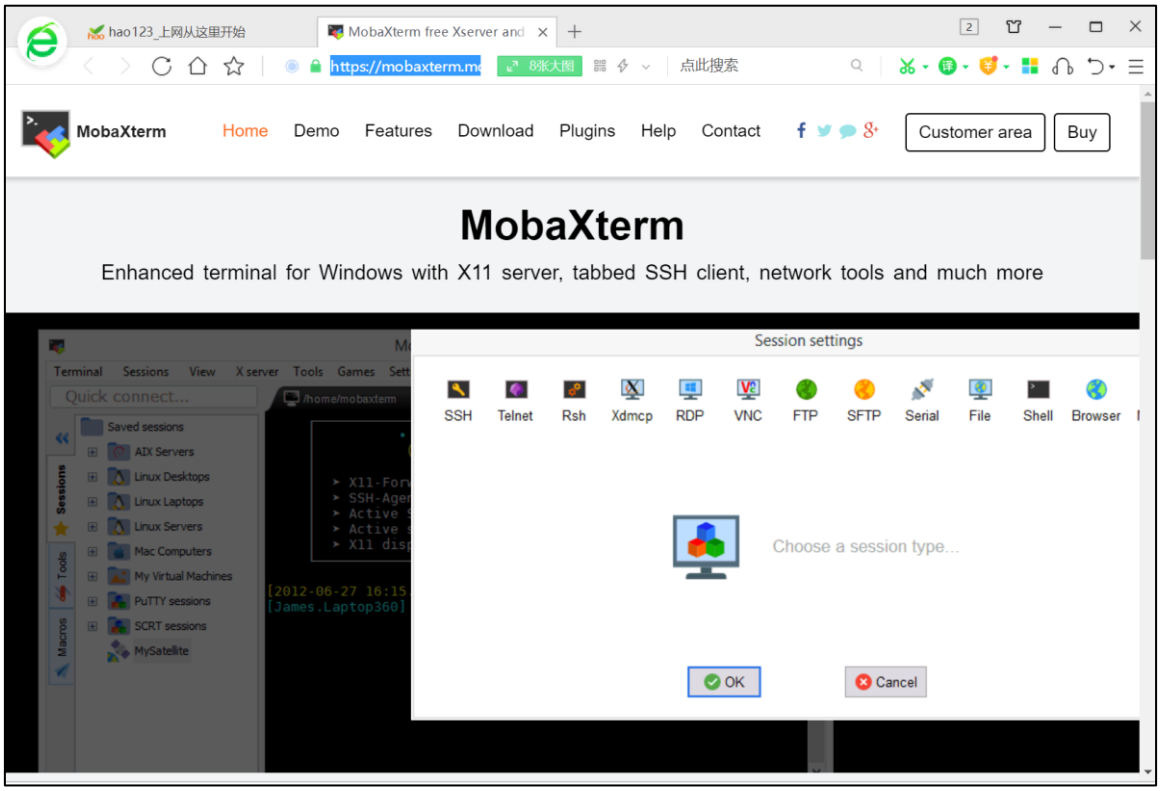



图 3.5.1.1 MobaXterm 官网

点击图 3.5.1.1 中的“Download”按钮即可打开下载界面，如图 3.5.1.2 所示：

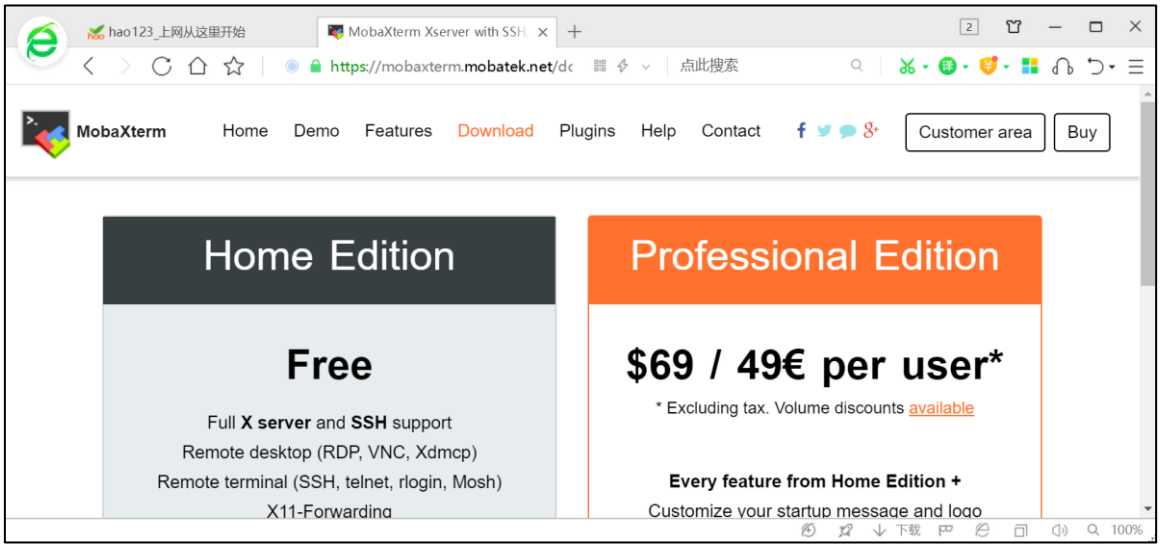



图 3.5.1.2 下载界面

从图 3.5.1.2 可以看出，一共有两个版本，左侧为免费的 Home Edition 版本，右侧为付费的 Professional Edition 版本。毫无疑问，我们肯定选择免费的 Home Edition 版，点击下方的“Download now”，打开下载界面，如图 3.5.1.3 所示：

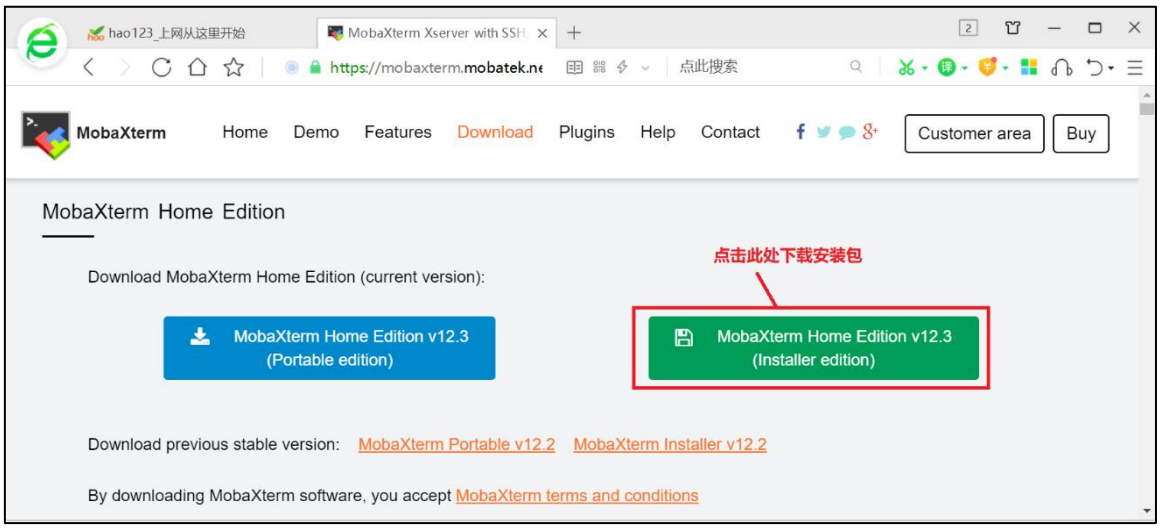



图 3.5.1.3 下载界面

可以看出，当前的版本号为 v12.3，点击右侧按钮下载安装包。安装包已经放到了开发板光盘中，路径为：开发板光盘->3、软件->MobaXterm_Installer_v12.3.zip。打开此压缩包，然后双击 MobaXterm_installer_12.3.msi 进行安装，安装方法很简单，一步一步进行即可。安装完成以后就会在桌面出现 MobaXterm 图标，如图 3.5.1.4 所示，如果桌面没有的话就自行添加。


图 3.5.1.4 MobaXterm 软件图标

### 3.5.2 MobaXterm 软件使用

双击 MobXterm 图标，打开此软件，软件界面如图 3.5.2.1 所示：

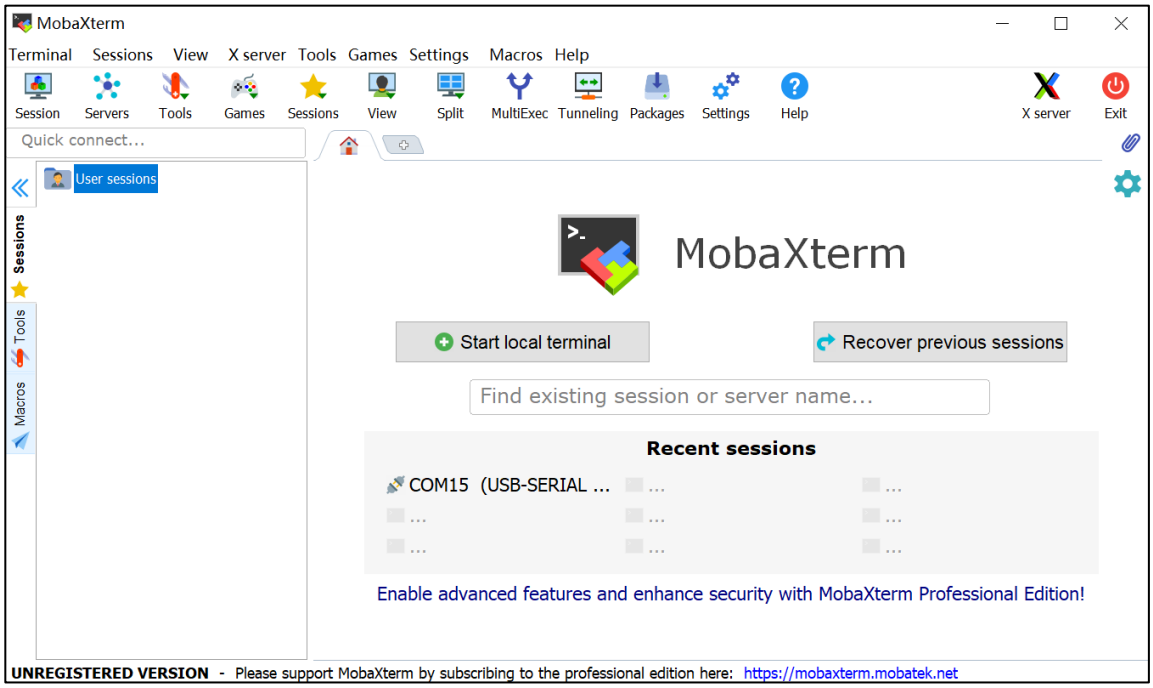



图 3.5.2.1 MobaXterm 软件主界面

点击菜单栏中的“Sessions->New session”按钮，打开新建会话窗口，如图 3.5.2.2 所示：

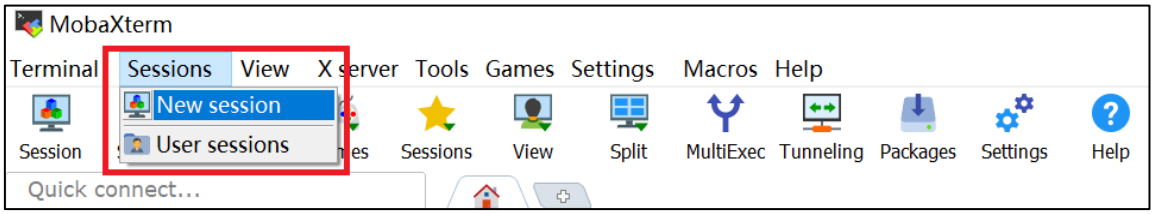



图 3.5.2.2 新建会话

打开以后的新建会话窗口如图 3.5.2.3 所示：

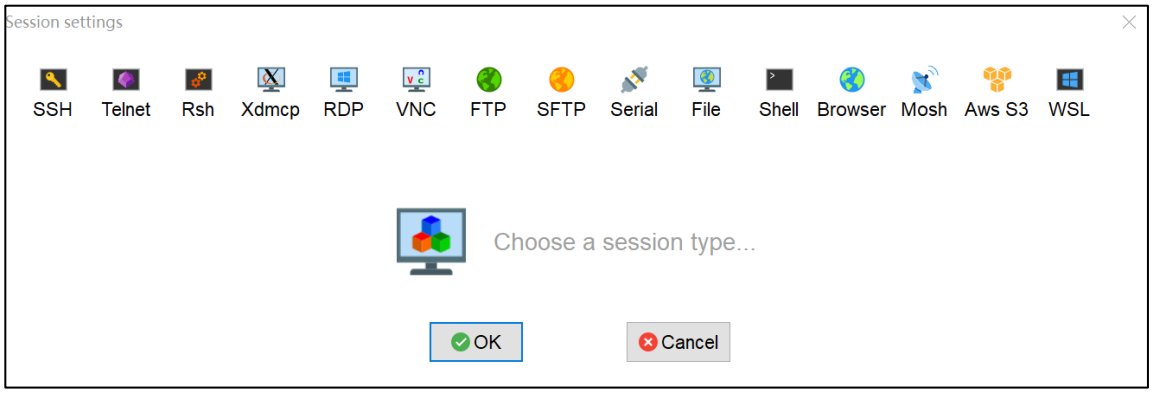



图 3.5.2.3 新建会话窗口

从图 3.5.2.3 可以看出，MobaXterm 软件支持很多种协议，比如 SSH、Telnet、Rsh、Xdmcp、RDP、VNC、FTP、SFTP、Serial 等等，我们现在就讲解一下如何建立 Serial 连接，也就是串口

连接，因为我们使用 MobaXterm 的主要目的就是作为串口终端使用。点击图 3.5.2.3 中的“Serial”按钮，打开串口设置界面，如图 3.5.2.4 所示：

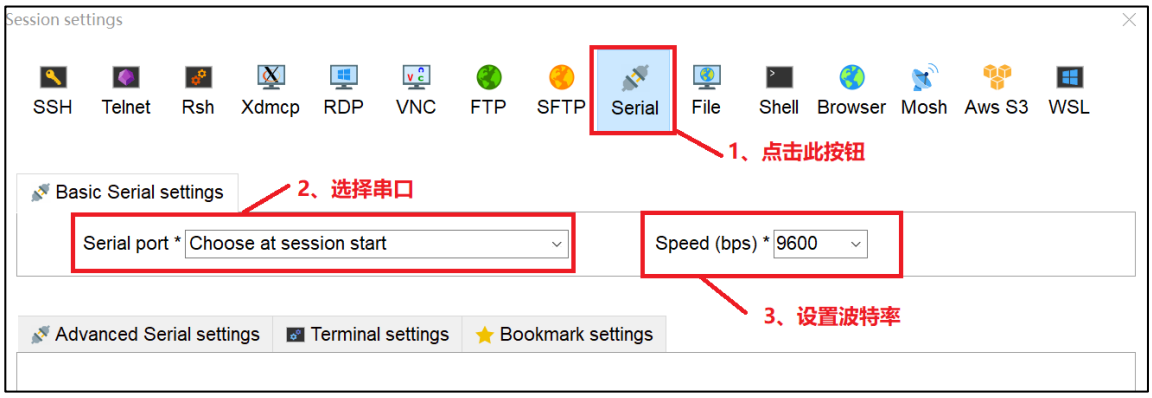



图 3.5.2.4 设置串口

打开串口设置窗口以后先选择要设置的串口号，因此要先用串口线将开发板连接到电脑上上，然后设置波特率为 1500000(根据自己实际需要设置)，完成以后如图 3.5.2.5 所示：


图 3.5.2.5 设置串口及其波特率

MobaXterm 软件可以自动识别串口，因此我们直接下拉选择即可，波特率也是同样的设置方式，下拉选择即可。完了以后还要设置串口的其他功能，下方一共有三个设置选项卡，如图 3.5.2.6 所示：


图 3.5.2.6 串口其他设置选项

点击 Advanced Serial settings 选项卡，设置串口的其他功能，比如串口引擎、数据位、停止位、奇偶校验和硬件流控等，按照图 3.5.2.7 所示设置即可：

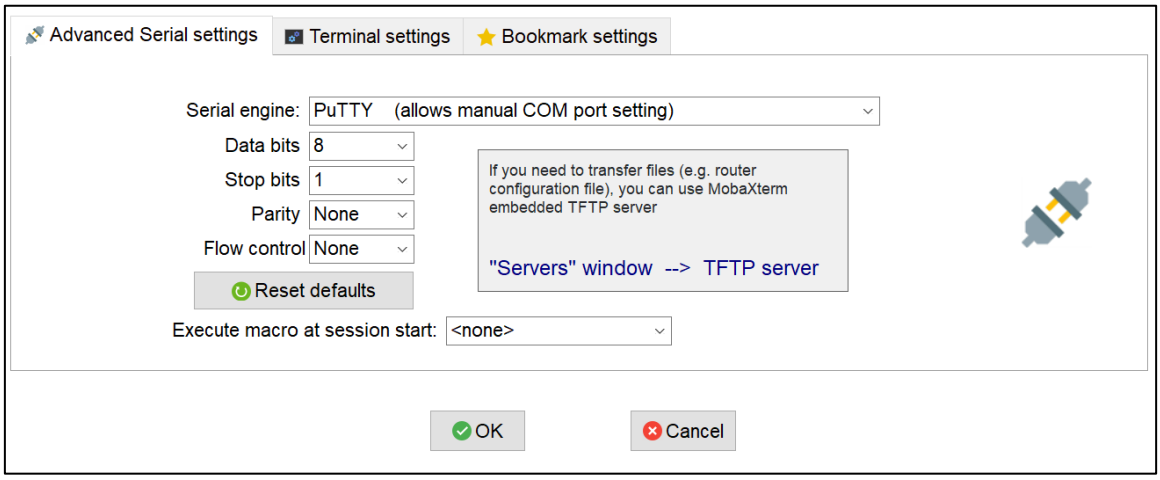



图 3.5.2.7 串口设置

如果要设置终端相关的功能的话点击“Terminal settings”即可，比如终端字体以及字体大小等。设置完成以后点击下方的“OK”按钮即可。串口设置完成以后就会打开对应的终端窗口，如图 3.5.2.8 所示：

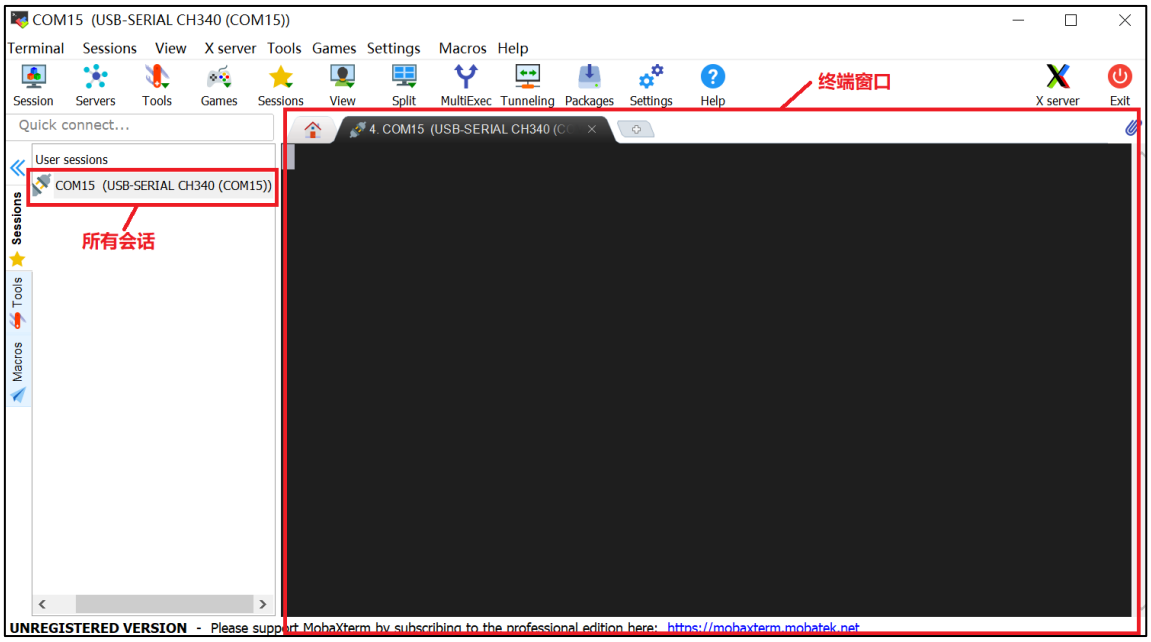



图 3.5.2.8 成功建立的串口终端

如果开发板里面烧写了系统的话就会在终端中打印出系统启动的 log 信息，如图 3.5.2.9 所示：

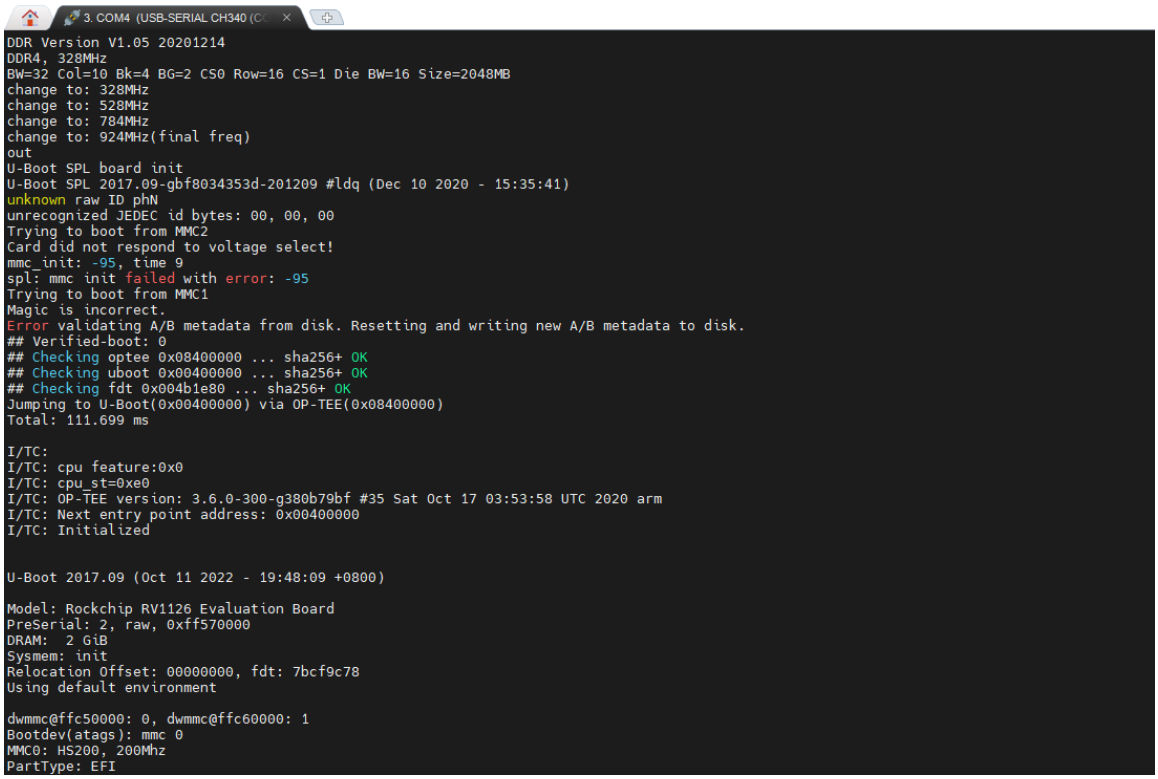



图 3.5.2.9 MobaXterm 作为串口终端

可以看出，MobaXterm 作为串口终端还是非常漂亮的，结合了 SecureCRT 的功能强大与 Putty 的免费。推荐大家使用 MobaXterm 作为串口终端使用，当然了，MobaXterm 也可以作为其他终端软件，这里大家就自行摸索吧。

## 3.6 ADB 的安装和使用

### 3.6.1 ADB 命令安装

ADB 命令的全称为“Android Debug Bridge”，从英文中看出主要是用作安卓的调试工具。ADB 命令在嵌入式开发中越来越常用了，在 RV1126 上 OTG 默认当作 ADB 功能(可以做复用其它功能)，所以我们要在 Windows 上安装 ADB 工具(linux 已经通过命令安装成功了)，安装包已经放到了开发板光盘，路径为：开发板光盘 A-基础资料→4、软件→ platform-tools_r33.0.3-windows.zip。解压到自定义的安装目录。接着我们在 Windows 上按“win”+“R”组合件打开运行，结果如下所示：

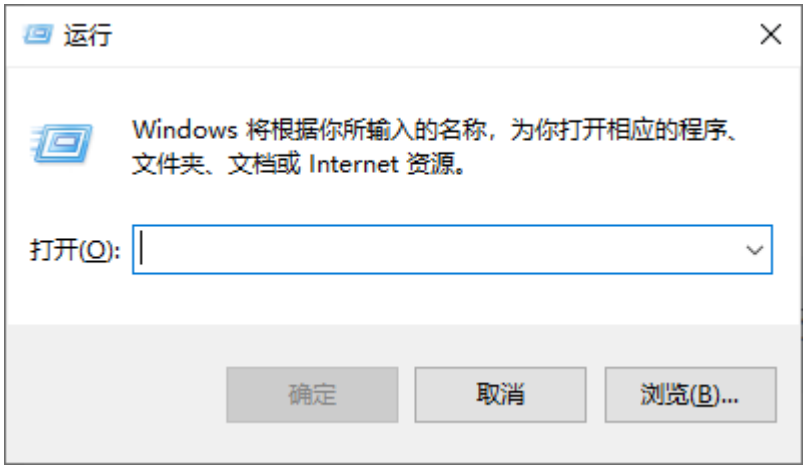



图 3.6.1.1 Windows10 的运行

打开运行后，输入 sysdm.cpl，按回车就会打开系统属性，如下图所示：

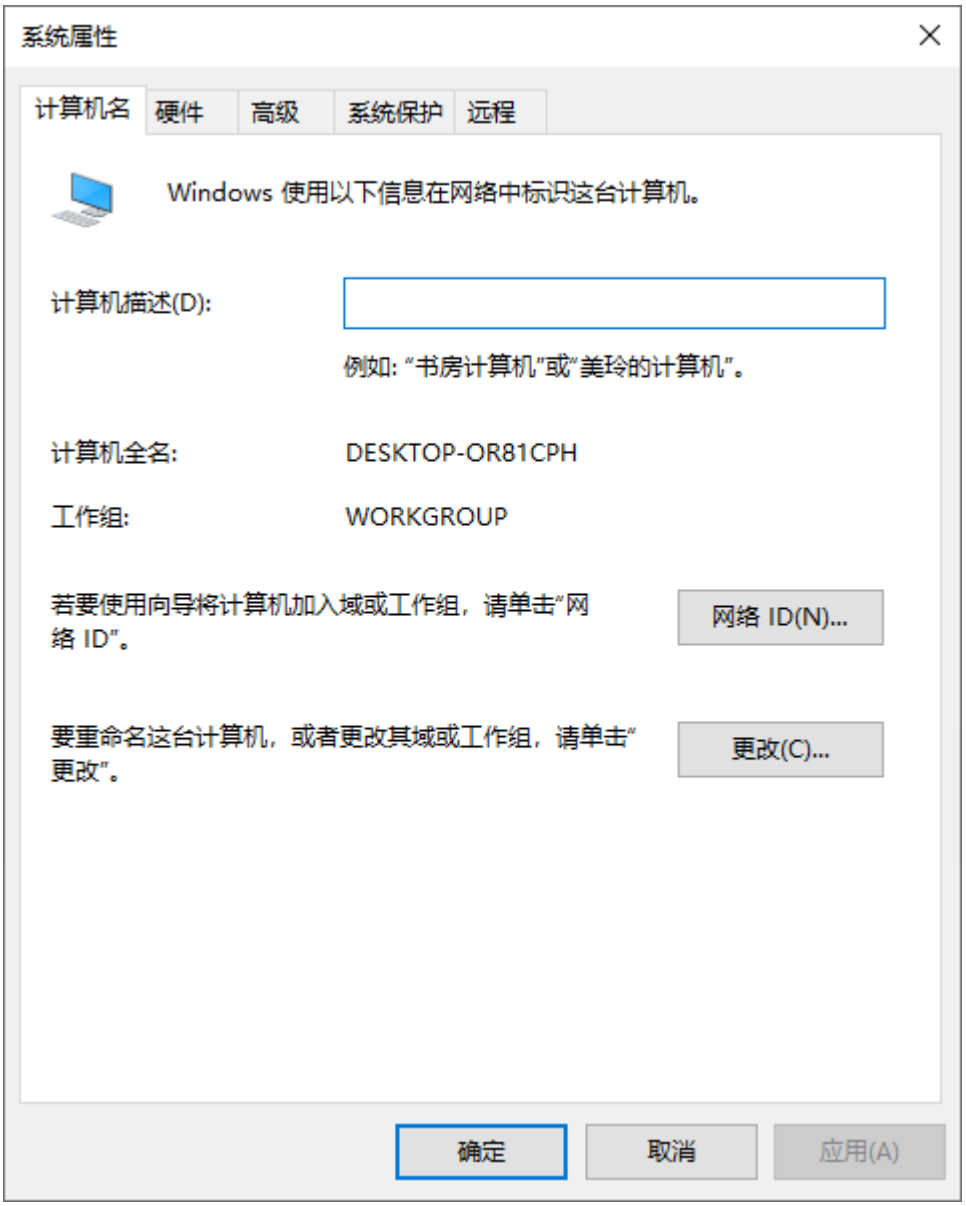



图 3.6.1.2 系统属性

点击图 3.6.1.2 中的“高级”，进入环境变量设置界面，如下图所示：

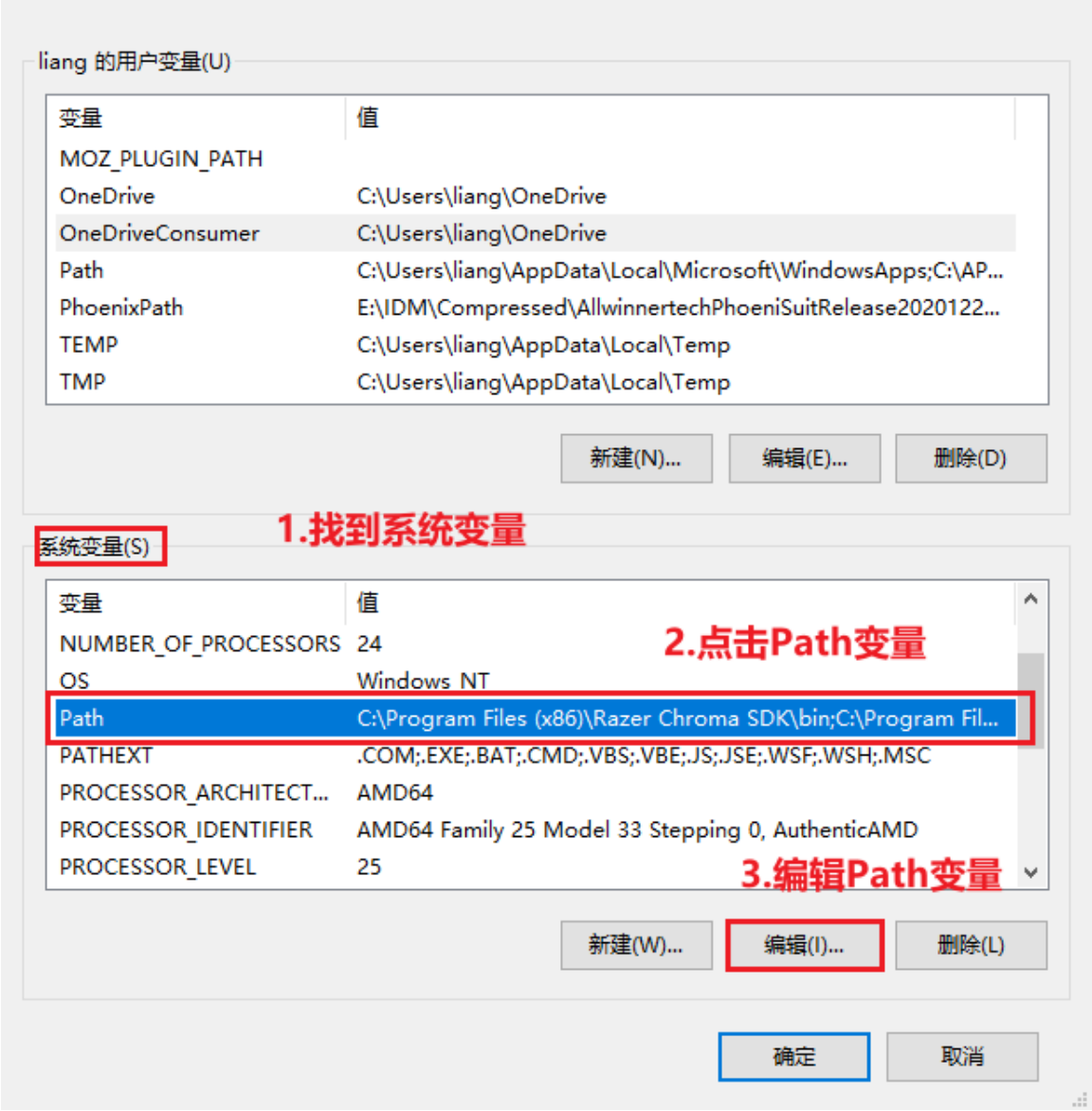



图 3.6.1.3 环境变量

接着我们可以把 ADB 的路径添加到系统变量里面，根据上图的步骤操作进入“Path”变量路径添加，如下图所示：

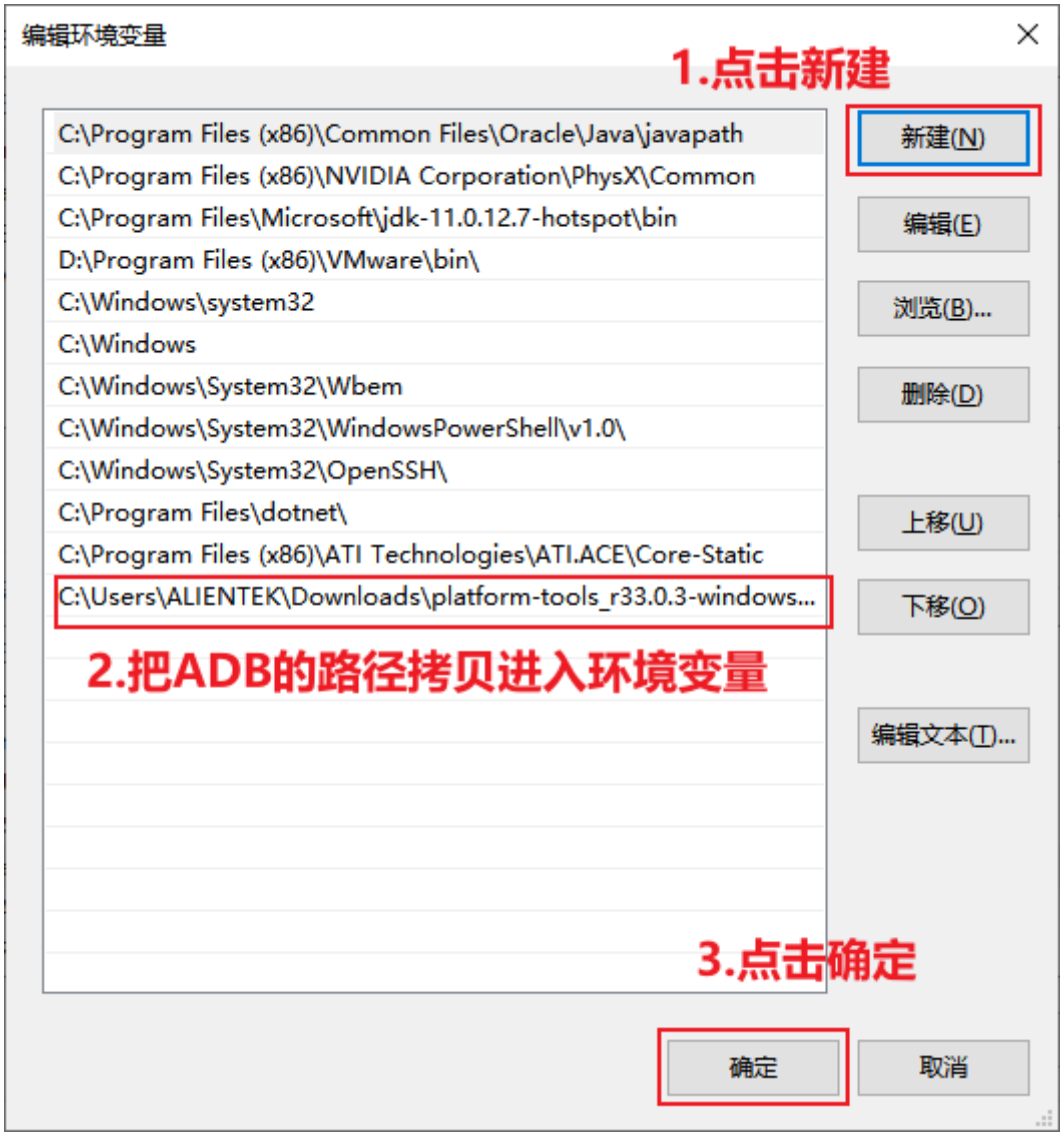



图 3.6.1.4 Path 编辑环境变量

根据上图 3.6.1.4 步骤把 ADB 的路径添加到 Path 系统环境变量里面(最好要点击两次确认)，这里笔者的路径为：C:\Users\ALIENTEK\Downloads\platform-tools_r33.0.3-windows\platform-tools。运行 CMD 终端，输入命令进行检验是否安装成功。命令如下所示：

```
adb --version
```

显示结果如下所示：

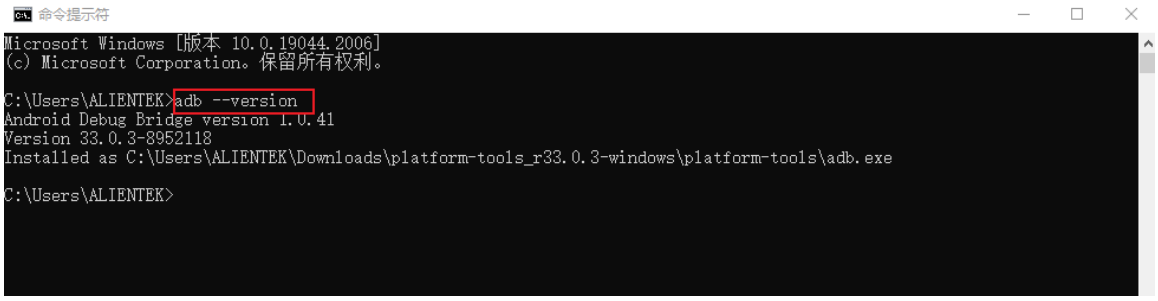



图 3.6.1.5 adb 版本验证

### 3.6.2 ADB 命令使用

这里笔者只列出 ADB 命令在嵌入式 Linux 下一些常用命令(adb 命令在 Windows 和 Linux 使用方法都是一样的)，这里就使用结合 ATK-DLRV1126 开发板和创建好的 Ubuntu 系统进行测试。首先我们先启动开发板(如果开发板没有系统请参考 3.7 小节进行烧录)，用 USB Type-C 线将开发板的 USB OTG 接口与电脑连接起来，连接方式如图所示：

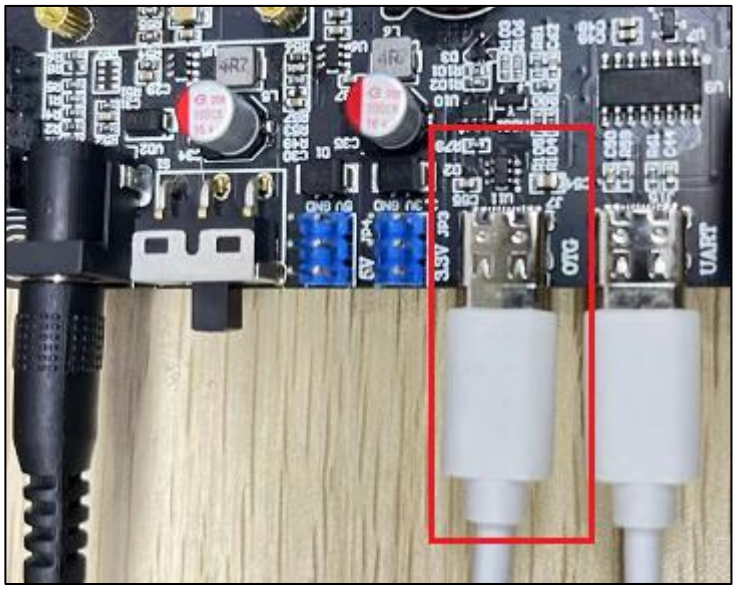


图 3.6.2.1 OTG 连接方式

默认情况下，USB 会连接到 Windwos 下，我们需要将 USB 连接到 Ubuntu，所以需要设置一下 VMware，WMware 右下角会有当前电脑所有连接的 USB 设备，鼠标放上去以后会显示每个 USB 设备的名字，我们找到含有“Fuzhou Rockchip Android ADB Interface”字样的 USB 设备，如图 3.6.2.2 所示：

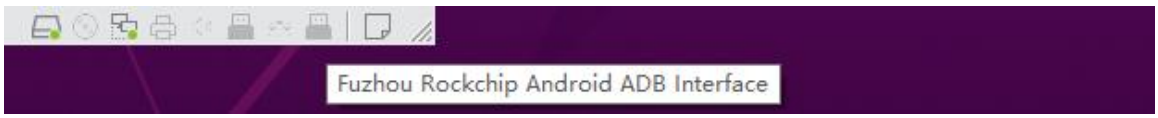



图 3.6.2.2 ATK-DLRV1126 USB ADB 接口

图 3.6.2.2 中第二个 USB 设备就是 ATK-DLRV1126 的 ADB 接口，此时图标是灰色的，说明并没有连接到 Ubuntu 下，需要我们进行设置,鼠标放到图 3.6.2.2 中 USB ADB 设备上，比如此时我的电脑就是第二个图标，鼠标放上去以后点击鼠标右键，结果如图 3.6.2.3 所示：

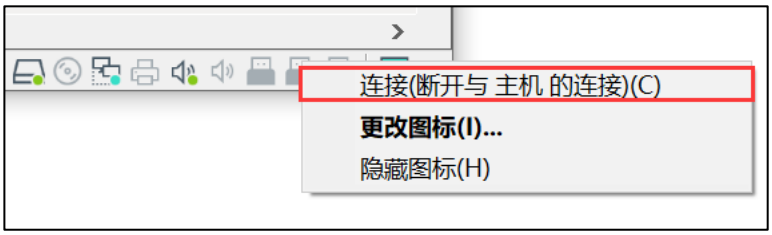



图 3.6.2.3 ADB 设备连接到虚拟机

点击图 3.6.2.3 中的“连接”按钮，此时 USB ADB 就会断开与主机(Windows)的连接，从而连接到虚拟机(Ubuntu)上。连接成功以后对应的 USB 图标颜色就会变深，如图 3.6.2.4 所示：

图 3.6.2.4 ADB 设备连接到虚拟机

连接后就可以使用 ADB 命令进行测试。ADB 命令的格式要求，如下文字所示：

**adb [-d|-e|-s <serialNumber>] <command>**

-d：指定当前唯一通过 USB 连接的 Android 设备为命令目标。

-e：指定当前唯一运行的模拟器为命令目标。

-s < serialNumber >：指定相应 serialNumber 号的设备/模拟器为命令目标。

command：对所需要设备执行命令。

- adb 帮助查看

```
adb help
```

此命令查看 ADB 帮助。结果如下图所示：

```
alientek@alientek-virtual-machine:~/桌面$ adb help
Android Debug Bridge version 1.0.39
Version 1:8.1.0+r23-5ubuntu2
Installed as /usr/lib/android-sdk/platform-tools/adb

global options:
 -a         listen on all network interfaces, not just localhost
 -d         use USB device (error if multiple devices connected)
 -e         use TCP/IP device (error if multiple TCP/IP devices available)
 -s SERIAL  use device with given serial (overrides $ANDROID_SERIAL)
 -t ID      use device with given transport id
 -H         name of adb server host [default=localhost]
 -P         port of adb server [default=5037]
 -L SOCKET  listen on given socket for adb server [default=tcp:localhost:5037]

general commands:
 devices [-l]             list connected devices (-l for long output)
 help                     show this help message
 version                  show version num

networking:
 connect HOST[:PORT]      connect to a device via TCP/IP [default port=5555]
 disconnect [HOST[:PORT]]
     disconnect from given TCP/IP device [default port=5555], or all
 forward --list           list all forward socket connections
 forward [--no-rebind] LOCAL REMOTE
```

图 3.6.2.5 adb 帮助命令

- 网络连接设备

```
adb connect <serialNumber>
```

注意：<serialNumber>表示要连接的设备，可以 IP 地址，比如: adb connect 192.168.6.118。(不建议用网络连接，最好使用 OTG 连接，把 OTG 和电脑连接上系统会自动连接)如果使用 USB 当作 ADB 连接系统会自动连接的，不能卸载。连接如下图所示：

```
alientek@alientek-virtual-machine:~/桌面$ adb connect 192.168.6.118
connected to 192.168.6.118:5555
alientek@alientek-virtual-machine:~/桌面$
```

图 3.6.2.6 adb 网络连接设备

- 查看连接设备

```
adb devices
```

此命令列出当前和计算机连接的 ADB 设备信息。结果如下图所示：

```
alientek@alientek-virtual-machine:~/桌面$ adb devices
List of devices attached
192.168.6.118:5555      device
336398dc82d8c3a5        device
```

图 3.6.2.7 ADB 连接设备

图 3.6.2.3 中看出有两个连接设备，192.168.6.118:5555 是通过网络连接的 ADB 设备。336398dc82d8c3a5 是通过 USB 连接的设备，device 表示已经连接上。

- 进入设备的 shell 终端

```
adb -s <serialNumber> shell
```

比如：adb -s 336398dc82d8c3a5 shell 进入 336398dc82d8c3a5 设备的 shell 终端里。如果只有一个设备可以直接使用 adb shell。进入终端后可以当作普通的终端设备操作开发板(相对于连接上串口)，退出终端可以直接输入 exit 即可结束设备终端操作，返回 Ubuntu 系统的终端。使用结果如下所示：

```
alientek@alientek-virtual-machine:~/桌面$ adb -s 336398dc82d8c3a5 shell
[root@RV1126_RV1109:/]#
[root@RV1126_RV1109:/]# ls
bin             init        media  proc           sdcard     udisk
busybox.config  lib         misc   rockchip_test  sys        userdata
data            lib32       mnt    root           test       usr
dev             linuxrc     oem    run            timestamp  var
etc             lost+found  opt    sbin           tmp        vendor
[root@RV1126_RV1109:/]# exit
alientek@alientek-virtual-machine:~/桌面$ adb -s 192.168.6.118:5555 shell
[root@RV1126_RV1109:/]# ls
bin             init        media  proc           sdcard     udisk
busybox.config  lib         misc   rockchip_test  sys        userdata
data            lib32       mnt    root           test       usr
dev             linuxrc     oem    run            timestamp  var
etc             lost+found  opt    sbin           tmp        vendor
[root@RV1126_RV1109:/]# exit
alientek@alientek-virtual-machine:~/桌面$
```

图 3.6.2.8 adb shell 终端

- ADB 网络断开连接

```
adb disconnect <serialNumber>
```

比如 adb disconnect 192.168.6.118:5555，断开网络设备 192.168.6.118:5555。结果如下所示：

```
alientek@alientek-virtual-machine:~/桌面$ adb disconnect 192.168.6.118:5555
disconnected 192.168.6.118:5555
alientek@alientek-virtual-machine:~/桌面$ adb devices
List of devices attached
336398dc82d8c3a5        device

alientek@alientek-virtual-machine:~/桌面$
```

图 3.6.2.9 断开 ADB 连接

- 拷贝文件到开发板

```
adb -s <serialNumber> push file /oem/
```

把 test 文件拷贝到“/oem”目录下，比如：adb -s 336398dc82d8c3a5 push test /oem/。操作结果如下所示：

```
alientek@alientek-virtual-machine:~/桌面$ adb -s 336398dc82d8c3a5 push test  /oem/
test: 1 file pushed. 1.7 MB/s (20971520 bytes in 11.528s)
alientek@alientek-virtual-machine:~/桌面$ adb -s 336398dc82d8c3a5 ls /oem/
000041ed 00000400 6344e1ba usr
000081ed 000009a1 633675ce RkLunch.sh
00008180 000000c3 00001701 .ash_history
000041ed 00000400 634403c5 etc
000081a4 0004d000 633675ce sysconfig-4K.db
000041ed 00001400 00000007 www
000081ed 000001d9 633675ce RkLunch-stop.sh
000041ed 00000400 00003a98 .
000081b6 01400000 634a57bb test
000041c0 00003000 6344e5af lost+found
000041ed 00001000 00000003 ..
000081a4 0004e000 633675ce sysconfig-2K.db
000081a4 0004d000 633675ce sysconfig-1080P.db
alientek@alientek-virtual-machine:~/桌面$
```

图 3.6.2.10 adb 拷贝文件

- 把文件拷贝到 PC 端

```
adb -s <serialNumber> pull /oem/test ./
```

把开发板里的“/oem/test”文件拷贝到当前路径里，比如：adb -s 192.168.6.118:5555 pull /oem/test ./ 。结果如下所示：

```
alientek@alientek-virtual-machine:~/桌面$ ls
vmware-tools-distrib
alientek@alientek-virtual-machine:~/桌面$ adb -s 192.168.6.118:5555 pull /oem/test ./
/oem/test: 1 file pulled. 2.6 MB/s (20971520 bytes in 7.833s)
alientek@alientek-virtual-machine:~/桌面$ ls
test  vmware-tools-distrib
alientek@alientek-virtual-machine:~/桌面$
```

图 3.6.2.11 adb 拷贝文件

当使用 adb devices 列出只有一个连接设备的时候，可以不用加-s < serialNumber >指定设备，默认就会使用设备。

## 3.7 瑞芯微开发工具的安装和使用

### 3.7.1 Rockchip 烧录驱动的安装

瑞芯微提供了 RKDevTool 上位机烧录工具，此工具只能在 Windows 系统下运行，运行前要先安装驱动文件。文件的路径为：开发板光盘 A-基础资料→04、软件→DriverAssitant_v5.0.zip，解压此文件。打开解压后的文件目录进入 DriverAssitant_v5.0\DriverAssitant_v5.0 目录。进入的目录如下图所示：

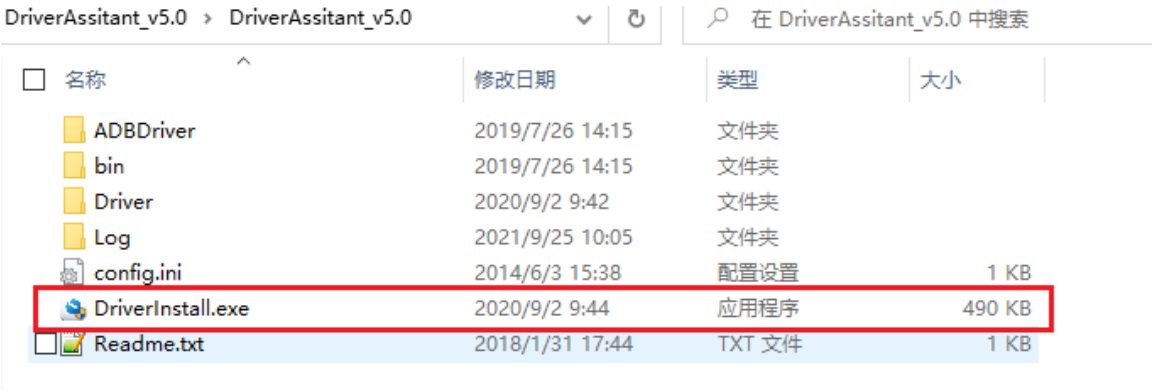



图 3.7.1.1 DriverAssitant_v5.0 目录

双击“DriverInstall.exe”就弹出一个节目点击驱动安装的按键，直接点击安装即可。结果如下所示：


图 3.7.1.2 瑞芯微烧录驱动安装

接着我们就可以使用 RKDevTool 软件了。文件路径为：开发板光盘 A-基础资料→04、软件→RKDevTool_Release_v2.81.zip，解压此文件。打开解压后的文件夹进入到如下界面：

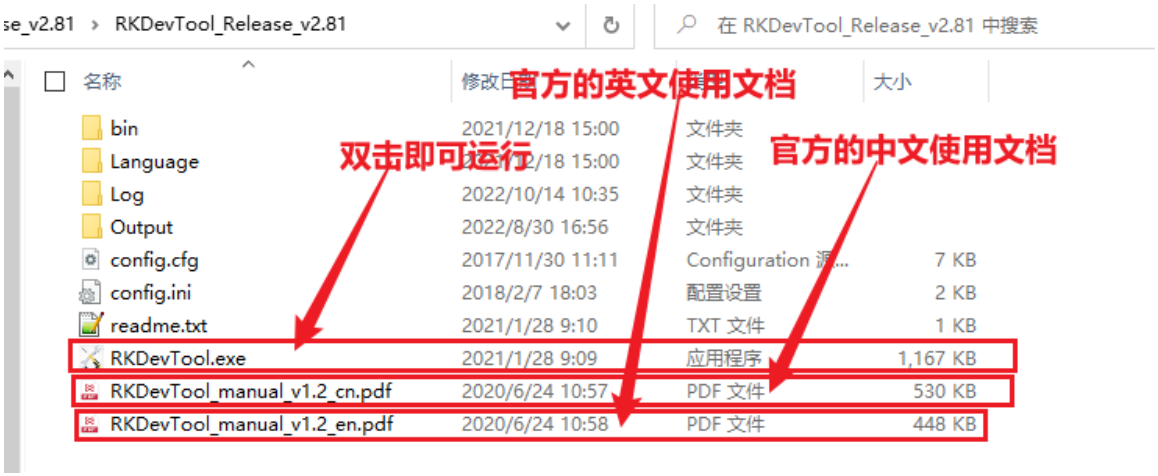



图 3.7.1.3 瑞芯微开发工具的文件图

点击图中 3.7.1.3RKDevTool.exe 文件即可运行。运行结果如下所示：

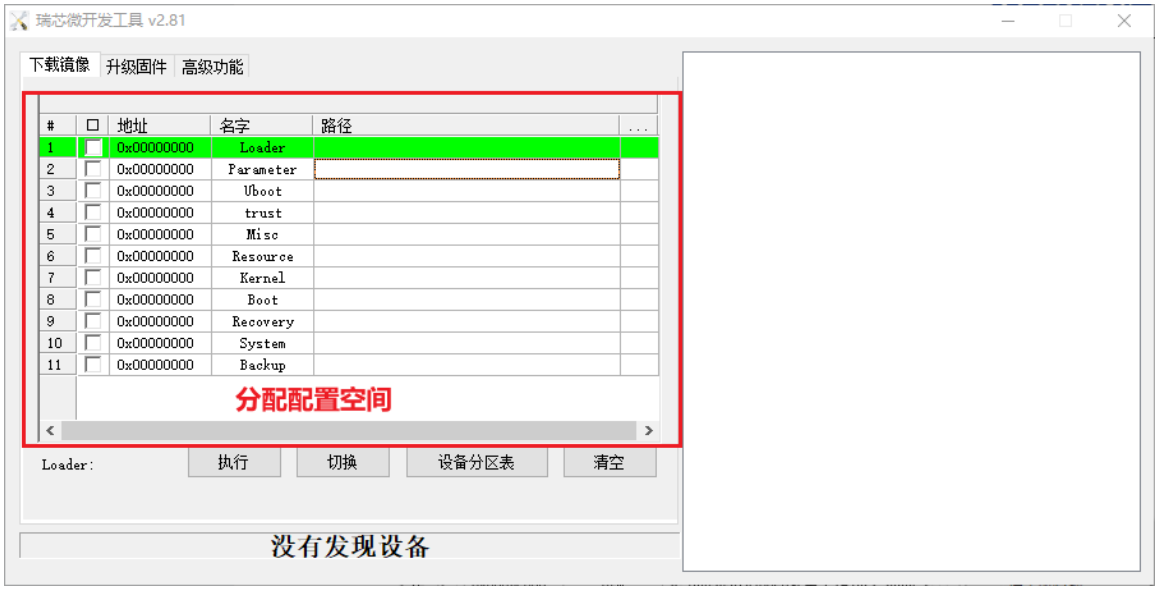



图 3.7.1.4 瑞芯微开发工具图

### 3.7.2 Rockchip 烧录工具使用

这里笔者只是简单说下如何使用烧录工具烧录 ATK-DLRV1126 开发板的出厂源码，如果想使用更多的功能请查看图 3.7.1.3 中的官方使用文档。在分区配置空间处右击鼠标，进入选择“导入配置”，如下图所示：


图 3.7.2.1 导入配置选项图

点击“导入配置”后就会弹出一个文件选择，支持的文件类型为“.cfg”。文件路径为：开发板光盘 A-基础资料→09、系统镜像→01、出厂系统 SDK 镜像→ ATK-DLRV1126 出厂系统配置.cfg。导入完成后如下图所示：

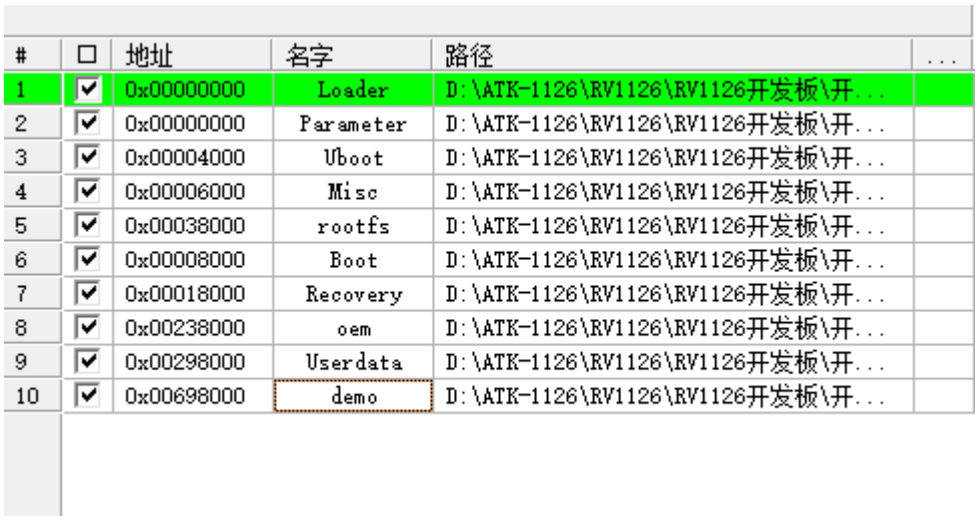



图 3.7.2.2 导入配置选项图


图 3.7.2.3 修改对应的文件路径

在图中 3.7.2.2 已经导入配置选项，可以看出来一共有 9 个烧录选项(为啥有 9 个选项是根

据 parameter 决定的)。“方框”里面打勾表示烧录，“地址”表示烧录到 emmc 地址，“名字”表示分区名字，“路径”表示要烧录到此选项的文件，“…”表示修改路径中的文件。图中的路径是笔者电脑上的，所以各位需要点击“…”去更改每一个选项的文件路径。每个烧录选项对应官方出厂系统的文件如下表所示：

| 烧录选项名字 | 出厂系统 SDK 镜像文件 |
|---|---|
| Loader | MiniLoaderAll.bin |
| Parameter | parameter.txt |
| Uboot | uboot.img |
| Misc | misc.img |
| Rootfs | rootfs.img |
| boot | boot.img |
| recovery | recovery.img |
| oem | oem.img |
| Userdata | userdata.img |
| dome | demo.img |

表 3.7.2.1 烧录镜像文件表

设置好自己的文件路径(设置好了记得导出配置选择，这样做不用每次进行烧录的时候选择文件，导出方法看图 3.7.2.1)。

- MASKROM 模式烧录

接上开发板电源和 OTG 接口再上电。接着按住“UPDATE”健，再按一下复位键进入“MASKROM”状态(最好多按几次复位键，有时候会误触)。操作结果如下图所示：


图 3.7.2.3 ATK-DLRV1126 MASKROM 烧录连接图

进入“MASKROM”状态如下图所示：

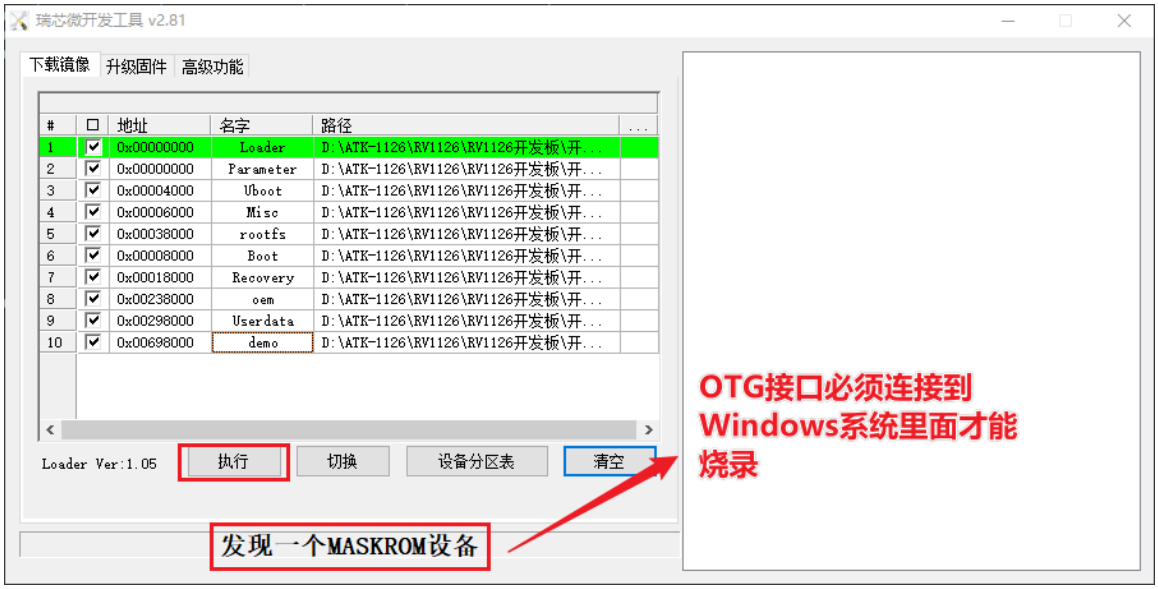



图 3.7.2.4 进入 MASKROM 模式图

接着点击执行即可进行烧录，烧录过程如下图所示：

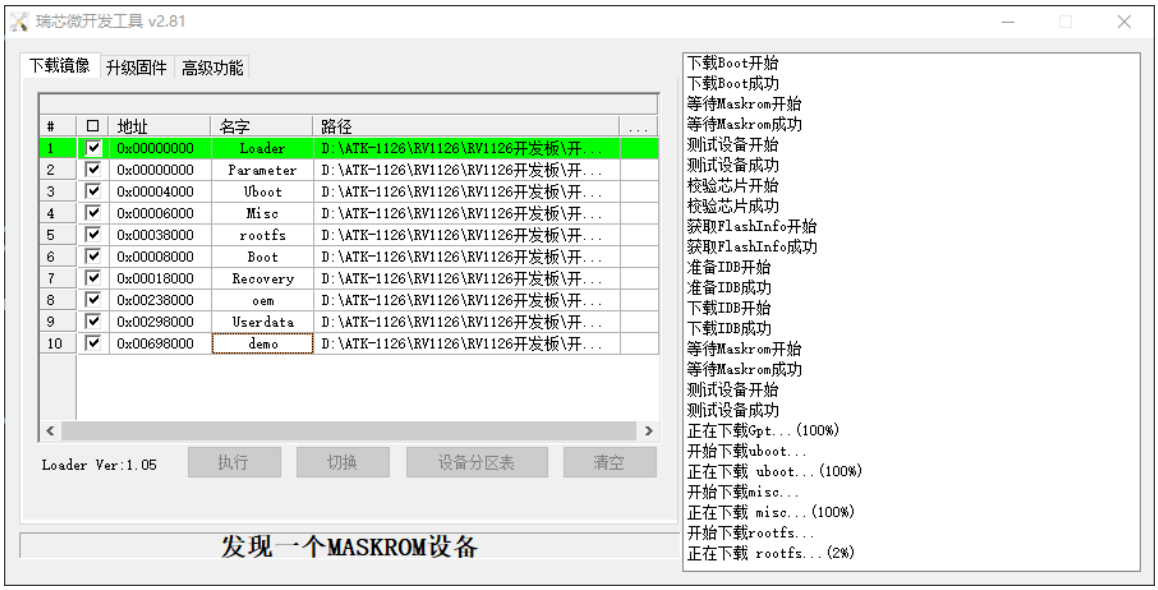



图 3.7.2.5 烧录状态图

图 3.7.2.5 中已经开始烧录了，烧录完成系统会自动启动。当开发板没有系统的时候，使用“MASKROM”模式进行出厂系统烧录。

当烧录失败说明你的 OTG 接口连接到 Ubuntu 系统里，如图所示：

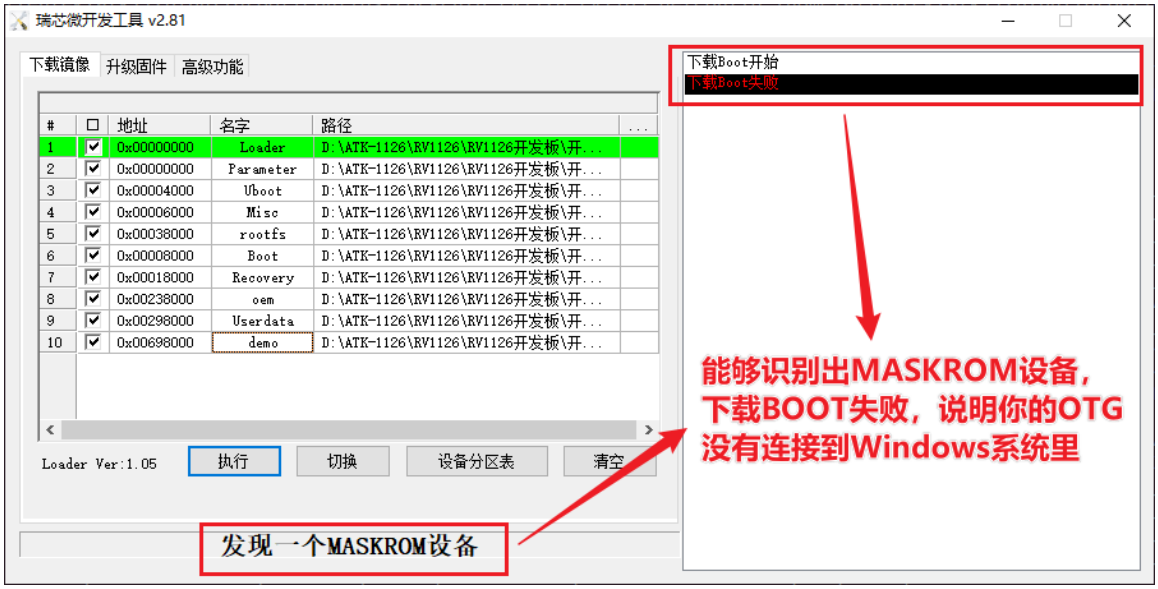



图 3.7.2.6 烧录固件失败

- LOADER 模式烧录

接上开发板电源和 OTG 接口再上电。接着按住“RECOVERY”健，再按一下复位键进入“LOADER”状态(最好多按几次复位键，有时候会误触)。LOADER 模式是使用 uboot 进行烧录的，所以要进入 LOADER 模式开发板必须能启动到 uboot 才能烧录。如果开发板是没有系统的会自动跳转到 MASKROM 模式，这边笔者已经通过 MASKROM 模式烧录了系统，所以可以进入 LOADER 模式，进入“LOADER”状态如下图所示：

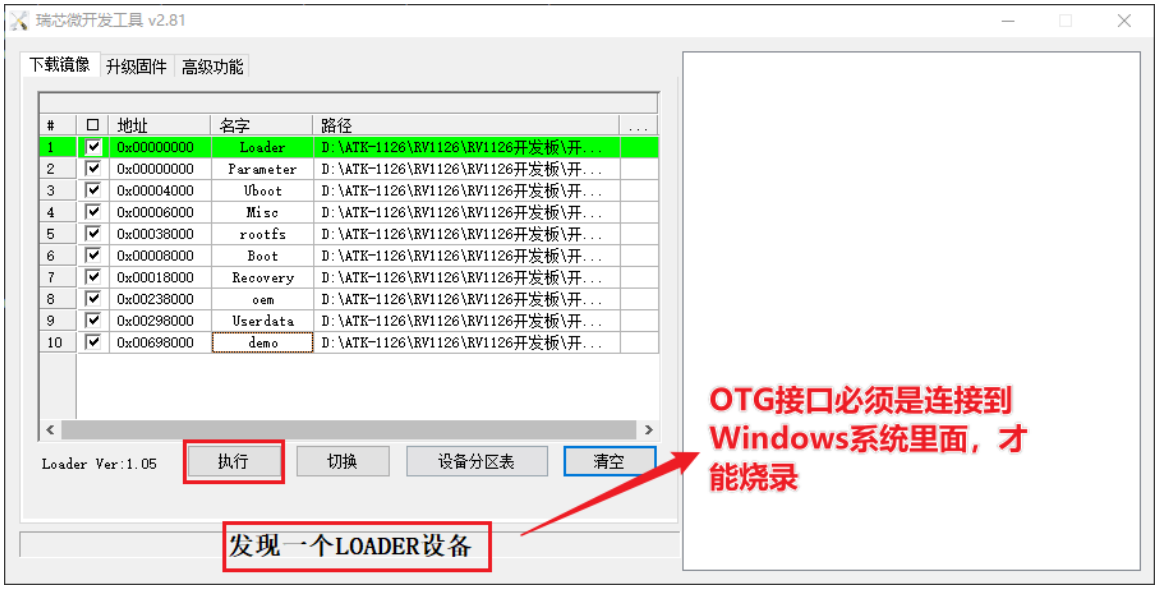



图 3.7.2.7 进入 LOADER 烧录图

接着点击执行即可进行烧录，烧录过程如下图所示：

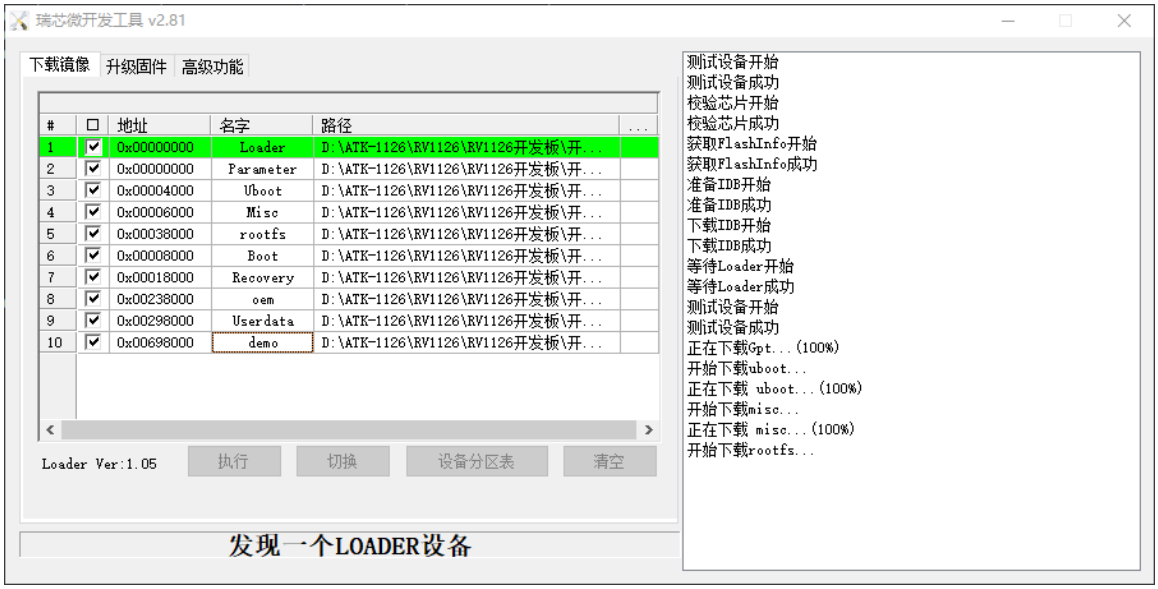



图 3.7.2.8 LOADER 烧录状态图

烧录完成后，系统会自动启动。

## 3.8 Update.img 包的烧录

在出厂系统的 SDK 镜像里面有一个 update.img 镜像，此镜像是根据 parameter.txt 文件生成一个镜像，把里面需要的文件全部打包到 update.img 里面。在 Windows 系统下烧录 update.img，此包的烧录支持 MASKROM 模式和 LOADER 模式。打开 Windows 的烧录工具，点击“升级固件”，结果如下图所示：

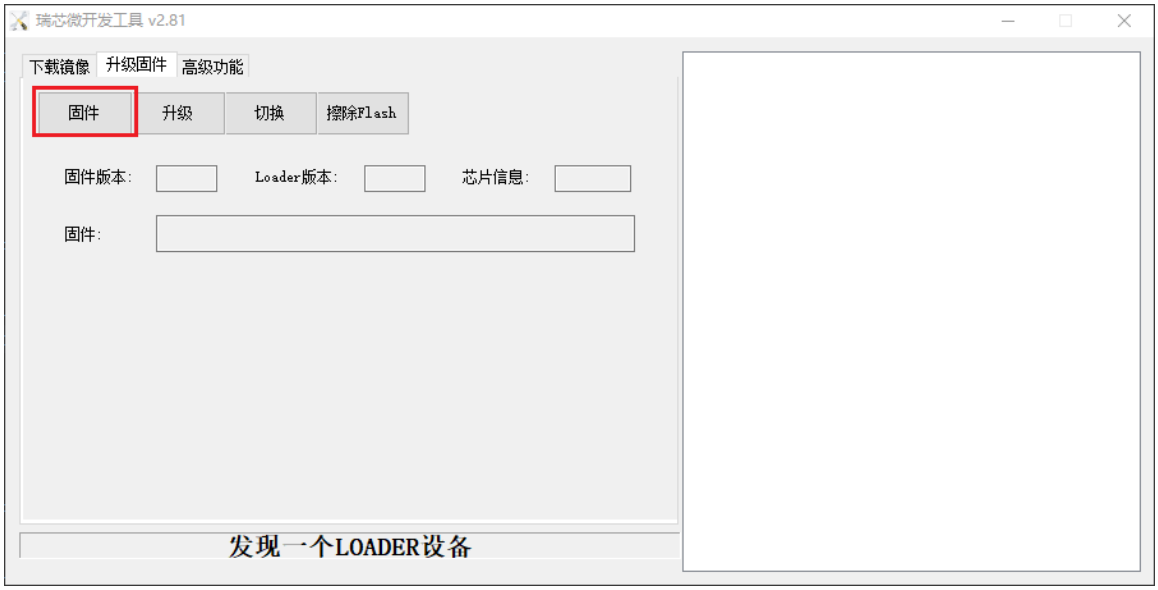



图 3.9.1 升级固件图

在上图中点击“固件”，即可加载 update.img 镜像，结果如下所示：

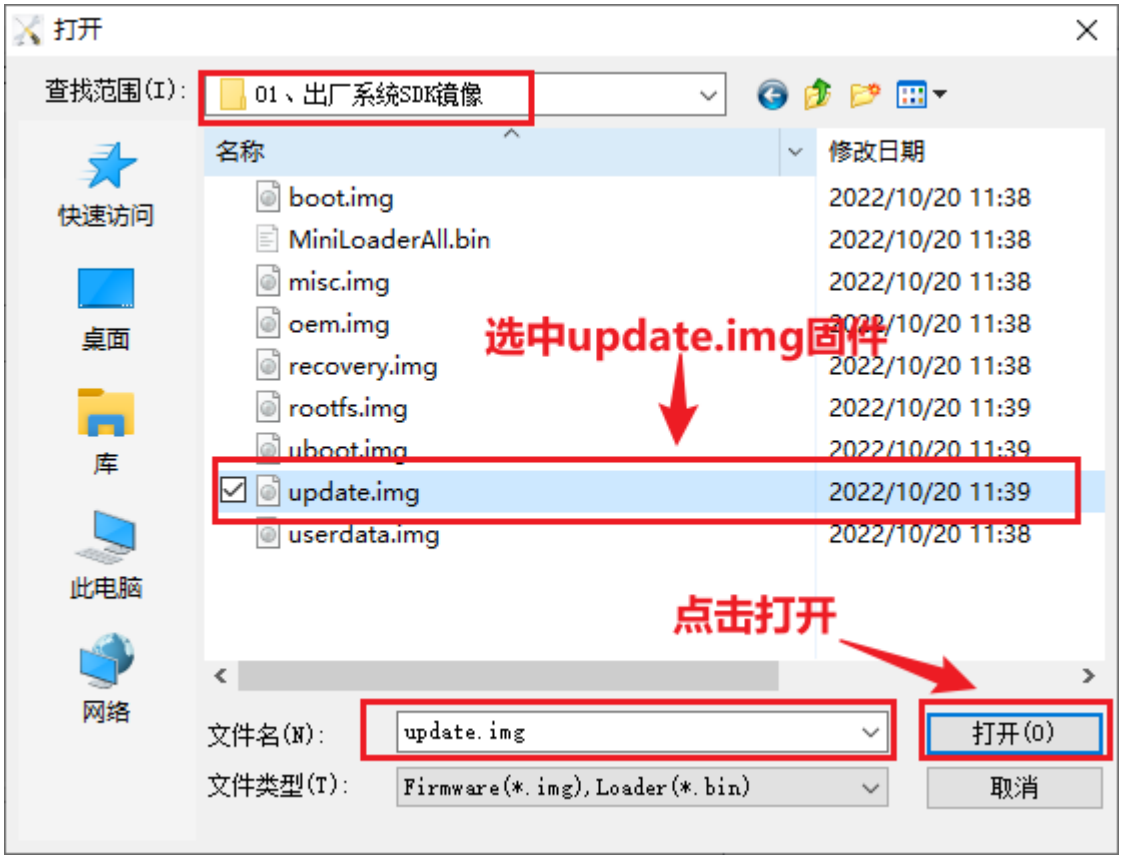



图 3.8.2 加载 update 固件

加载成功后如下图所示：


图 3.8.3 成功加载固件

有了固件后，我们可以点击“升级”把加载的 update.img 烧录到 emmc 里面去，点击“擦除”Flash 也可以把整个 EMMC 擦除。必须是加载固件后才能做这些操作。

## 3.9 Ubuntu 系统下烧录 ATK-DLRV1126 系统

本章节是通过 SDK 包自带的烧录工具进行烧录，先看完第 4 章节。在第 4 章节里面已经编译出整个 SDK 包所需要的文件了，先进入“LOADER”或者“MASKROM”模式，把 OTG 接口挂载到 Ubuntu 系统下，再跳转到源码目录下运行以下代码进行烧录：

```
sudo ./rkflash.sh //运行此命令是整个 SDK 烧录
```

运行结果如下所示：

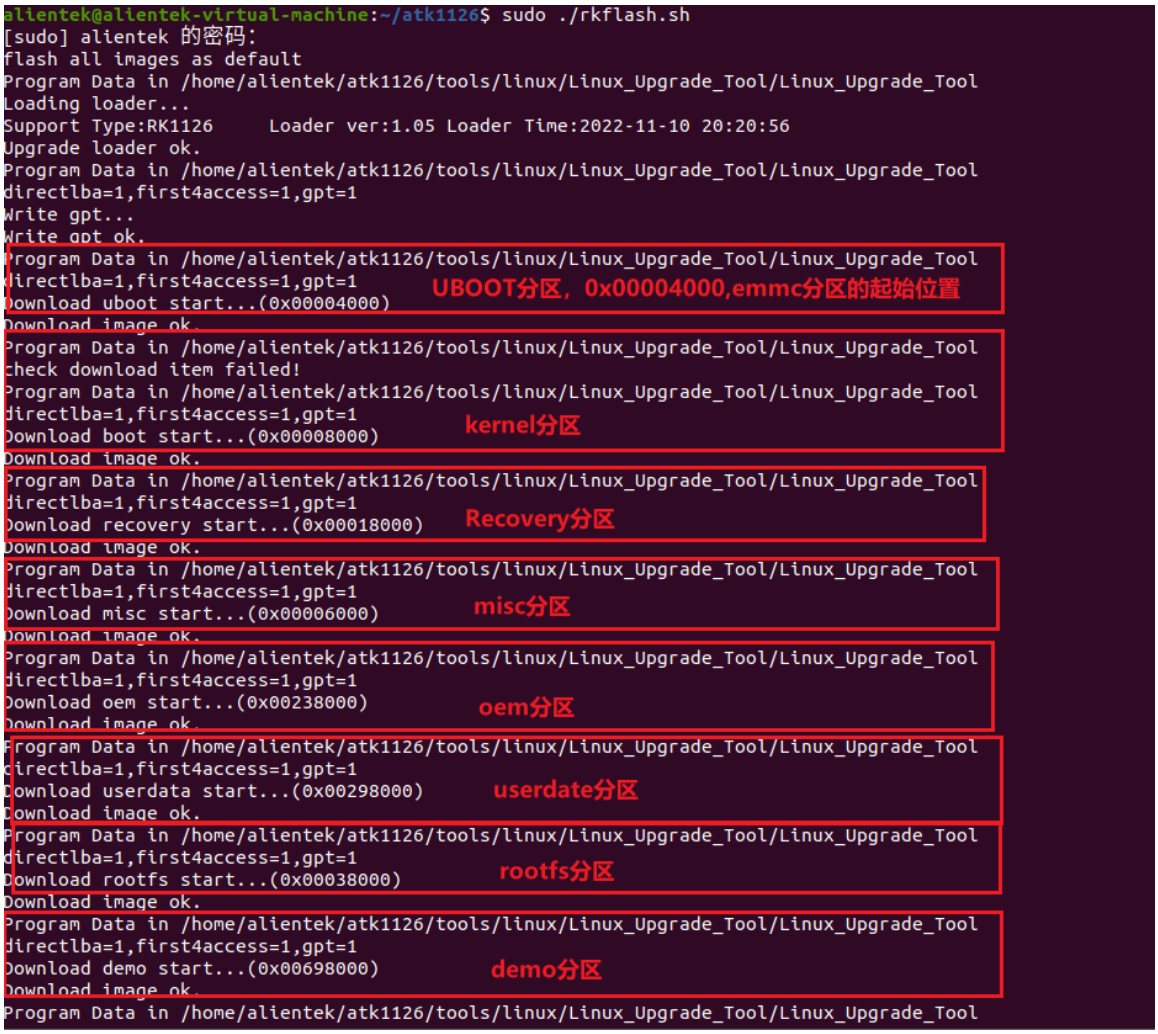



图 3.9.1 烧录官方 SDK 包

也可以运行 rkflash.sh 脚本进行单独烧录，开发板要先进入 LOADER 模式下(MASKROM 单独烧录很麻烦要先烧录 loader，所以直接进入 loader 模式就行了)。因为这个模式已经启动到 uboot，在烧录的时候可以使用 uboot 命令单独烧录到对应的分区。命令如下表格所示：

| 命令 | 说明 |
|---|---|
| sudo ./rkflash.sh loader | 烧录 loader 分区 |
| sudo ./rkflash.sh parameter | 烧录 parameter 文件 |
| sudo ./rkflash.sh recovery | 烧录 recovery 分区 |
| sudo ./rkflash.sh uboot | 烧录 uboot 分区 |
| sudo ./rkflash.sh boot | 烧录 boot 分区 |

| | |
|---|---|
| sudo ./rkflash.sh misc | 烧录 misc 分区 |
| sudo ./rkflash.sh oem | 烧录 oem 分区 |
| sudo ./rkflash.sh userdata | 烧录 userdata 分区 |
| sudo ./rkflash.sh rootfs | 烧录 rootfs 分区 |
| sudo ./rkflash.sh update | 烧录 update.img，两个烧录模式都能用 |

表格 3.9.1 LOADER 单独烧录命令

## 3.10 安装交叉编译工具链

### 3.10.1 拷贝交叉编译工具链

编译 SDK 是比较花时间的，为了方便大家不需要编译 SDK 就能直接编译 AI 例程来进行测试，正点原子专门定制了一套交叉编译工具链，安装包位于开发板光盘 A→05、开发工具→01、交叉编译工具→ atk-dlrv1126-toolchain-arm-buildroot-linux-gnueabihf-x86_64.run，如下图 3.10.1 所示：

注意：此交叉编译工具链后期还会更新，还不是最终版，更新的目的就是为了适配更多的例程，如需重新安装，直接卸载再安装最新版本的即可，安装的过程很简单。

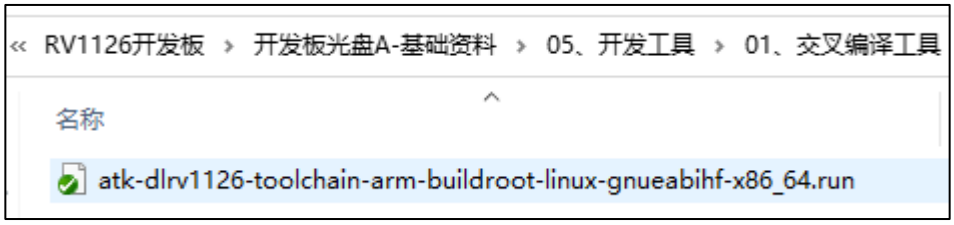



图 3.2.1 叉编译工具链

将 atk-dlrv1126-toolchain-arm-buildroot-linux-gnueabihf-x86_64.run 拷贝到 Ubuntu 下，如下图 3.2.2 所示，笔者拷贝到了 Ubuntu 的家目录下了。

```
alientek@alientek-virtual-machine:~$ ls atk-dlrv1126-toolchain-arm-buildroot-linux-gnueabihf-x86_64.run -l
-rw------- 1 alientek alientek 711078951 12月  5 15:04 atk-dlrv1126-toolchain-arm-buildroot-linux-gnueabihf-x86_64.run
alientek@alientek-virtual-machine:~$
```

图 3.2.12 拷贝好的叉编译工具链

拷贝完成后，记得用 ls -l 命令检查文件的属性是否是可执行的，上图中，笔者拷贝完成后，此文件已经具有可执行权限，可直接运行，若检查没有可执行权限，记得执行如下命令设置为可执行权限。

```
chmod a+x atk-dlrv1126-toolchain-arm-buildroot-linux-gnueabihf-x86_64.run
```

### 3.2.2 安装交叉编译工具链

执行如下命令直接安装交叉编译工具链，安装过程如下图 3.2.2.1 所示。

```
./atk-dlrv1126-toolchain-arm-buildroot-linux-gnueabihf-x86_64.run
```

当提示“Enter target directory for toolchain (default: /opt/atk-dlrv1126-toolchain):”时，表示是否选择默认安装在/opt/atk-dlrv1126-toolchain 目录下，建议直接选择默认安装路径，直接按下回车键即可。当提示“You are about to install the toolchain to "/opt/atk-dlrv1126-toolchain". Proceed[Y/n]?”时，直接按下“Y”即可。当弹出提示“$ . export PATH=$PATH:/opt/atk-dlrv1126-toolchain/usr/bin”时，表示已经安装完成。

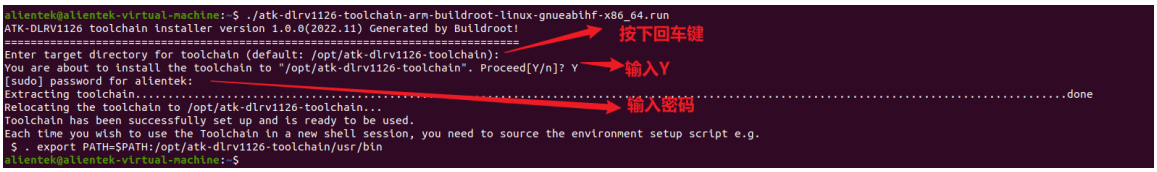



图 3.2.2.1 安装叉编译工具链

当安装完成后，在/opt 目录下就可以看到安装目录，如下图 3.2.2.2 所示，atk-dlrv1126-toolchain 下就是本次安装的交叉编译工具链的目录，而 st 目录是我以前安装 STM32MP157 的交叉编译工具链的目录。

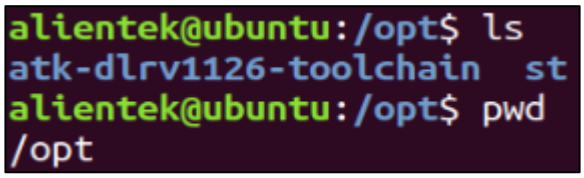



图 3.2.2.2 叉编译工具链安装目录

可以进入/opt/atk-dlrv1126-toolchain/bin 目录下，大概看一下，如下图 3.2.2.3 所示有不少 arm-linux-gnueabihf-*文件，说明交叉编译工具链初步安装完成，若要确定是否已经成功安装，我们可以尝试编译一个 AI 例程即可。

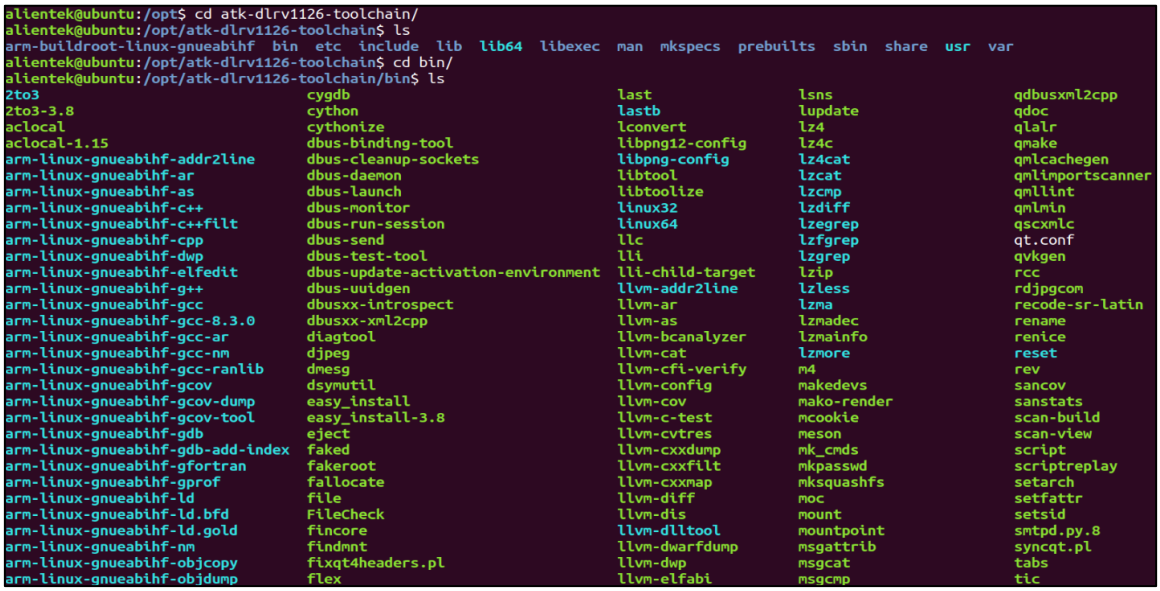
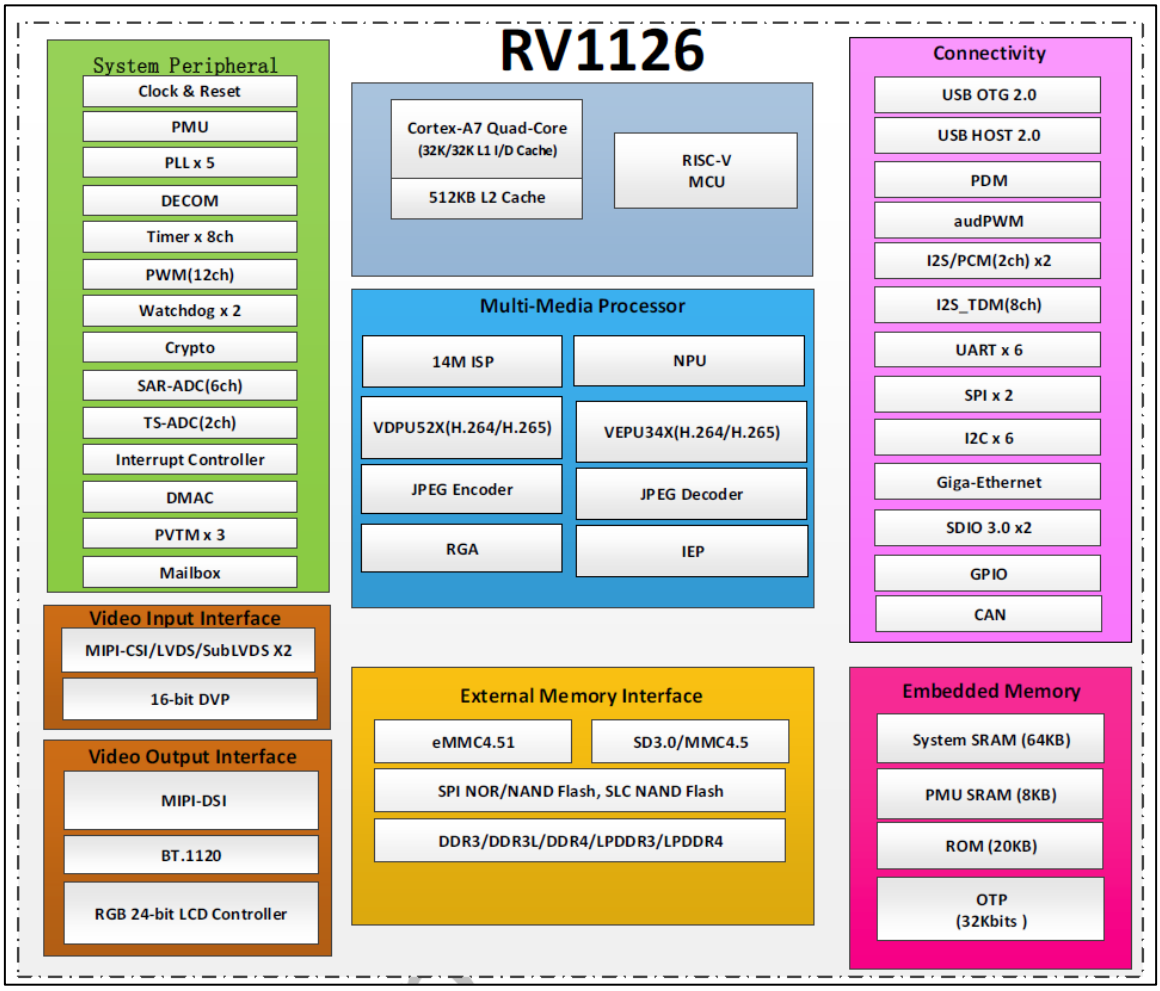



图 3.2.2.2 叉编译工具链安装目录

# 第四章 SDK 包的使用

ATK-DLRV1126 开发板具有高性能低功耗，是一款基于 Rockchip RV1126 芯片开发的开发板。RV1126 是 Rockchip 推出的一款编解码芯片，CPU 为 4 核 ARM Cortex-A7 32 位，专用于面向人工智能的机械视觉领域，支持 4K 编解码，支持 8 路 1080P 同时进行编解码，内置 2.0TOPS 的 NPU。像这种专用芯片，芯片厂商都会给出芯片的 SDK 包进行二次开发，用它来开发有很多好处，比如：不用安装一下特殊的软件和库，自带交叉工具链、文件系统和第三方库，可以直接编译出系统镜像，开发环境搭建比较容易等等。但是开发自由度不够高，更新软件版本很麻烦，代码里有很多私货(芯片厂商自己定义的代码)等等，所以本章节就讲解如何使用和编译 SDK 包。下图是 RV1126 芯片框架图，如下图所示：



## 4.1 SDK 包源码简要

首先建议大家在编译 SDK 之前，先到正点原子资料网站下载 ATK-DLRV1126 开发板最新的 SDK 包，地址为：http://www.openedv.com/docs/boards/arm-linux/RV1126%20Linux.html，点击下载链接进入下载网站，如下图所示：


图 4.1.1 ATK-DLRV1126 下载链接图

在上图红色框住的为 SDK 下载链接地址，进入百度云下载完整的 SDK 包，下完成后如下图所示：

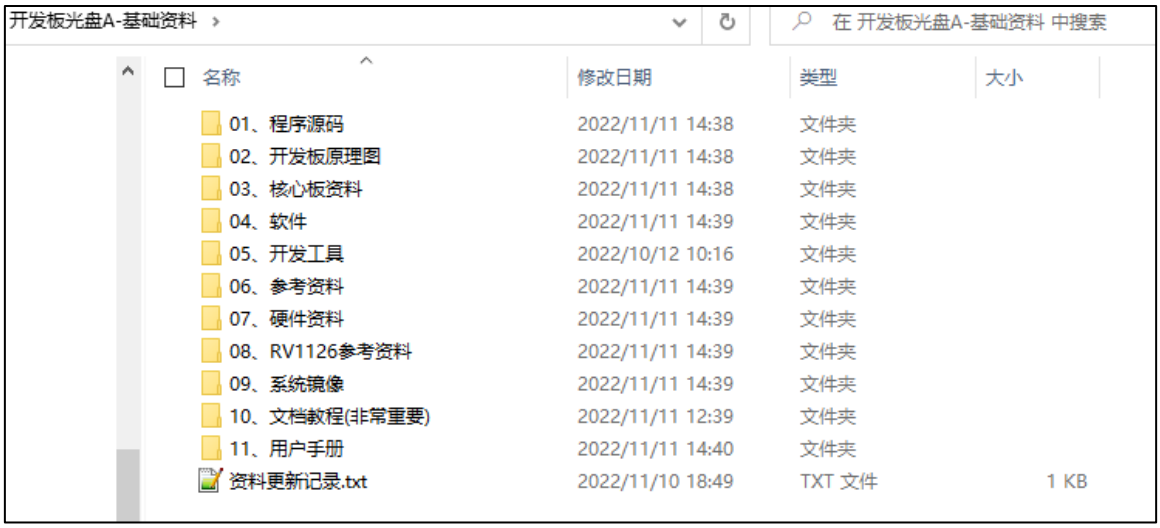



图 4.1.2 SDK 完整目录图

进入 01、程序源码→01、正点原子 SDK 源码目录，里面有一个 atk-rv1126_linux_release_v1.1_20221207.tar.bz2 文件，此文件是正点原子维护的 RV1126 SDK 包，把此包拷贝到 Ubuntu 系统里面，在 Ubuntu 下解压出来，进入工程目录有下图的文件目录，如下图所示：


图 4.1.1 SDK 包目录

从上图可以看出有很多文件夹，我们讲解一下每个目录或者文件的作用：

- app：存放上层应用程序的目录。
- buildroot ：SDK 包使用的文件系统为 buildroot。
- build.sh ：编译用的脚本，使用方法后面会教。
- device/rockchip ：存放每个平台的一些编译和打包固件的脚步和预备文件。
- docs ：存放 RK 开发指导文件、平台支持列表、工具使用文档、Linux 开发指南等。
- envsetup.sh ：要修改文件系统时候要设置的环境脚本。
- external：存放相关的库，包括音频，视频等。
- kernel ：kernel 源码。
- makefile ：整个 SDK 包编译的 Makefile。
- mkfirmware.sh ：固件打包使用的脚本，默认在当前路径下的 rockdev 目录。
- prebuilts：存放交叉编译工具链。
- rkbin: 存放固件和工具。
- rkflash.sh : linux 下的系统烧录脚本。
- tools ：存放固件和工具的目录。
- u-boot ：U-boot 源码目录。
- rockdev：存放编译输出固件的目录(整个 SDK 包编译完成后就会创建)。

### 4.1.1 rv1126 模块代码目录相关说明

什么叫做模块代码？一个完整的 SDK 包除了 kernel、u-boot、buildroot 之外，还需要提供上层的第三方库和 APP，第三方库和 APP 合起来叫做模块代码。像阿尔法开发板或者 ATK-CLMP157B 开发板只有 3 座大山(kernel、u-boot 和 buildroot)，没有模块代码，这样开发起来很麻烦。有了模块代码后，我们做产品就很容易了。比如：在 rv1126 上做人脸识别可以参考 rockface 模块代码。也可以做监控摄像头可以参考 common_algorithm、ipc-daemon、ipcweb-backend 和 ipcweb-ng 等等。一个 SDK 包包含了很多产品的应用代码。

| 部分模块代码目录路径 | 模块功能描述 |
|---|---|
| external/linux-rga | Raster Graphic Acceleration (RGA2) |
| external/recovery | recovery 和 Rockchip 升级代码 |
| external/rkwifibt | Wi-Fi 和 BT |
| external/rk_pcba_test | PCBA 测试代码 |
| external/isp2-ipc | 图像信号处理服务端 |
| external/mpp | 编解码代码 |
| external/rkmedia | Rockchip 多媒体封装接口 |
| external/camera_engine_rkaiq | 图像处理算法模块 |
| external/rknpu | NPU 驱动 |
| external/rockface | 人脸识别代码 |
| external/CallFunIpc | 应用进程间通信代码 |
| external/common_algorithm | 音视频通用算法库 |

| | |
|---|---|
| external/rknn-toolkit | 模型转换、推理和性能评估的开发套件 |
| app/libIPCProtocol | 基于 dbus，提供进程间通信的函数接口 |
| app/mediaserver | 提供多媒体服务的主应用（用于 IPC 应用开发参考或简单功能演 示） |
| app/ipc-daemon | 系统守护服务 |
| app/dbserver | 数据库服务 |
| app/netserver | 网络服务 |
| app/storage_manager | 存储管理服务 |
| app/ipcweb-backend | web 后端 |
| app/librkdb | 数据库接口 |
| app/ipcweb-ng | web 前端，采用 Angular 8 框架 |
| app/minigui_demo | 基于 MiniGUI 实现一个简单画图 demo |

表 4.1.1.1 rv1126 模块功能表格

### 4.1.2 rv1126 开发相关文档说明

本小节就列出官方开发文档说明：(默认的路径为：开发板光盘 A-基础资料→08、rv1126 参考资料)。

摄像头的驱动开发文档为：RV1126_RV1109→Camera→Rockchip_Driver_Guide_VI_CN_v1.0.8.pdf。

后续更新……

## 4.2 SDK 包下的脚本使用

在 4.1 小节讲解 SDK 包的源码目录下有 4 个重要的脚本，本小节就教大家如何使用这些脚本进行编译。

- build.sh 脚本使用

瑞芯微官方使用 build.sh 脚本来控制整个 SDK 包的编译和打包镜像，可以使用“-h”或者“help”命令来查看支持那些参数。运行以下命令即可查看：

```
./build.sh lunch
```

第一运行的时候必须要指定板级的配置信息(瑞芯微很多芯片都是基于此 SDK 进行开发的，所以要告诉 SDK 包我们使用什么芯片)，运行结果如下：

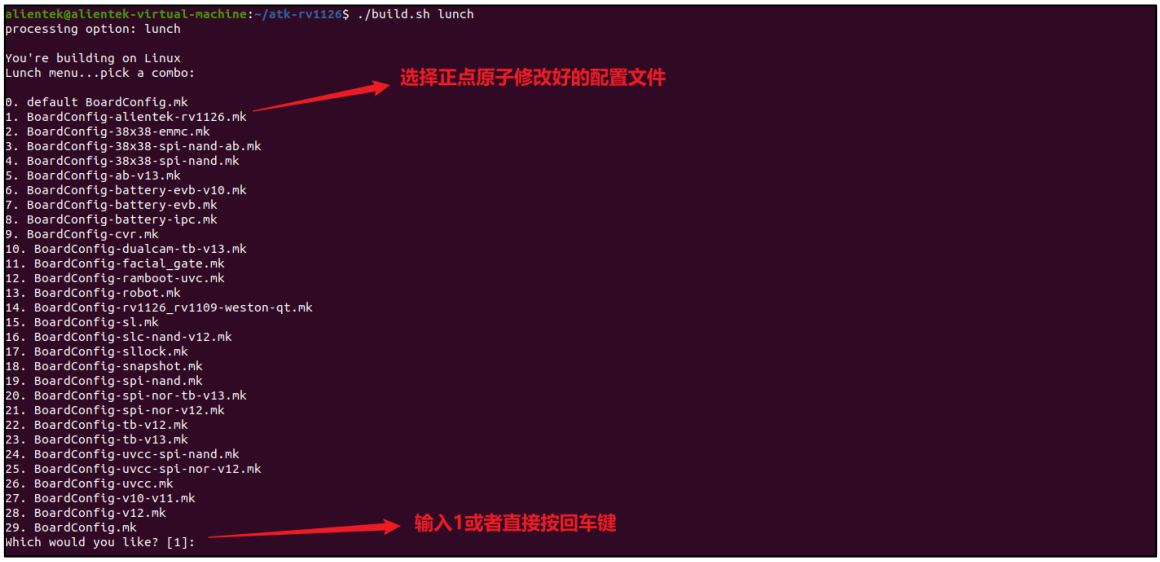



图 4.2.1 板级配置文件选择图

查看 build.sh 使用帮助，命令如下所示：

```
./build.sh -h //或者./build.sh help
```

运行结果如下所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ ./build.sh -h
Usage: build.sh [OPTIONS]
Available options:
BoardConfig*.mk     -switch to specified board config
lunch               -list current SDK boards and switch to specified board config
uboot               -build uboot
spl                 -build spl
loader              -build loader
kernel              -build kernel
modules             -build kernel modules
toolchain           -build toolchain
rootfs              -build default rootfs, currently build buildroot as default
buildroot           -build buildroot rootfs
ramboot             -build ramboot image
multi-npu_boot      -build boot image for multi-npu board
yocto               -build yocto rootfs
debian              -build debian10 buster/x11 rootfs
distro              -build debian10 buster/wayland rootfs
pcba                -build pcba
recovery            -build recovery
all                 -build uboot, kernel, rootfs, recovery image
cleanall            -clean uboot, kernel, rootfs, recovery
firmware            -pack all the image we need to boot up system
updateimg           -pack update image
otapackage          -pack ab update otapackage image (update_ota.img)
sdpackage           -pack update sdcard package image (update_sdcard.img)
save                -save images, patches, commands used to debug
allsave             -build all & firmware & updateimg & save
check               -check the environment of building
info                -see the current board building information
app/<pkg>           -build packages in the dir of app/*
external/<pkg>      -build packages in the dir of external/*

Default option is 'allsave'.
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 4.2.2 帮助命令参数

上图可以 build.sh 支持很多参数，不是所有参数都可以使用的。在 rv1126 上比较常用的参数有：uboot、lunch、kernel、all、buildroot 和 recovery 等等。下面我们说下常用的参数，如下表格所示：

| build.sh 参数 | 说明 | 例子 |
|---|---|---|
| BoardConfig*.mk | 选择板级的配置文件 | ./build.sh device/rockchip/rv1126_rv1109/BoardConfig-alientek-rv1126.mk |

| | | |
|---|---|---|
| lunch | 列出支持的板级配置文件，再选择板级的配置文件 | ./build.sh lunch |
| uboot | 编译 uboot | ./build.sh uboot |
| kernel | 编译 kernel | ./build.sh kernel |
| modules | 编译内核模块 | ./build.sh modules |
| rootfs | 编译文件系统 | ./build.sh rootfs |
| recovery | 编译 recovery | ./build.sh recovery |
| all | 编译整个 SDK 模块代码包 | ./build.sh all |
| cleanall | 清除整个 SDK 包 | ./build.sh cleanall |
| firmware | 打包系统镜像 | ./build.sh firmware |
| updateimg | 打包 update 镜像 | ./build.sh updateimg |
| app/<pkg> | 编译 app 里面的模块代码 | ./build.sh app/ipc-daemon |
| external/<pkg> | 编译 external 里面的模块代码 | ./build.sh external/rkmedia |

表 4.2.1 build.sh 参数表格

- envsetup.sh 脚本使用

envsetup.sh 脚本主要的作用是，使能 buildroot 的配置文件。在 RV1126 这个芯片里，一共可以选择的配置文件有 3 种分别为：文件系统的配置文件，recovery 分区的配置文件(此分区主要是用作升级和复原的文件系统，它也是一个文件系统)和 libs 的配置文件(生成一些库，方便我们写应用层代码做测试，导出文件系统的库文件)。运行命令如下所示：

```
./envsetup.sh
```

运行结果如下图所示：

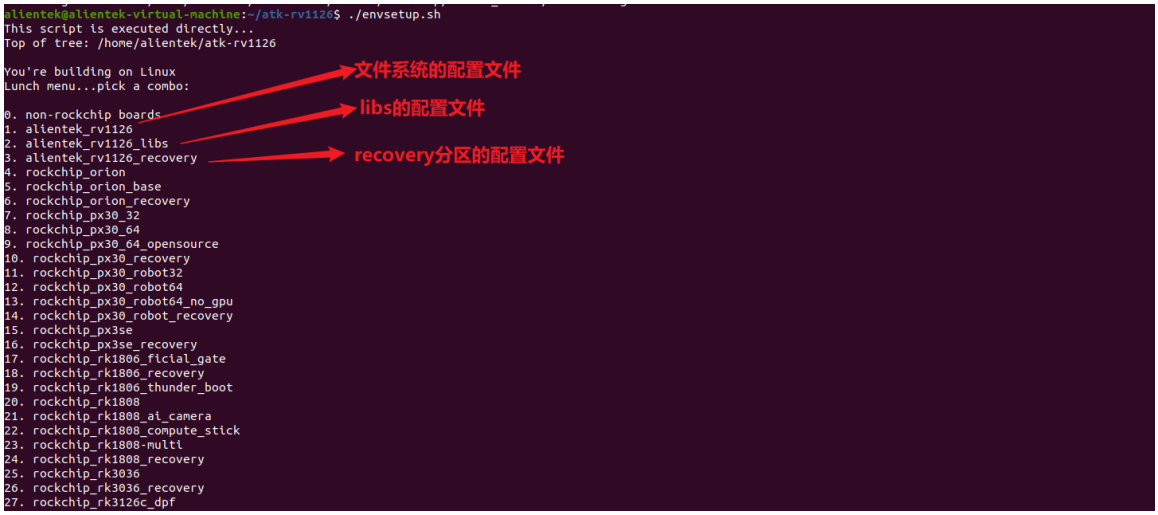



图 4.2.3 envsetup 的运行结果

Buildroot 的输出目录下只能有一个“.config”文件，所以我们配置不同的如果我们要修改文件系统的配置文件，需要先使用./envsetup.sh 选择 1 这个配置，就会在输出目录下生成“.config”文件，运行结果如下图所示：

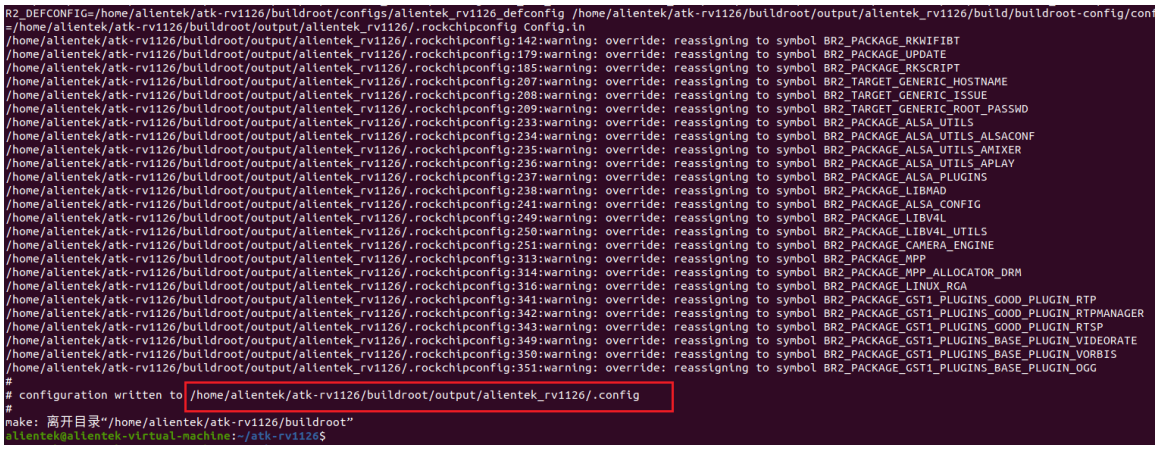



图 4.2.4 选择文件系统的配置

图中红色框的内容“/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/.config”是配置文件的绝对路径，此时的配置文件只能配置文件系统，不能配置 recovery 和 libs。使能完成后，在源码目录下可以当作 buildroot 的源码目录，比如我们可以进行文件系统的图形配置文件，运行以下代码即可：

```
make menuconfig
```

如下图所示：

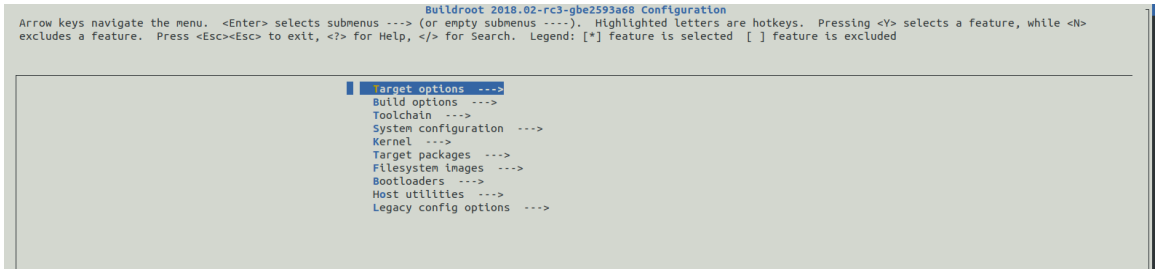



图 4.2.5 图形化配置文件

- mkfirmware.sh 脚本使用

mkfirmware.sh 脚本主要的作用是，把 uboot、kernel、文件系统等等进行打包和生成镜像。使用方法请看 4.11 小节。

- rkflash.sh 脚本使用

rkflash.sh 脚本主要的作用是，在 linux 系统下进行固件的烧录，使用方法请看 3.9 小节。

## 4.3 板级文件说明

BoardConfig-alientek-rv1126.mk 是正点原子为 ATK-DLRV1126 开发板添加的板级信息，在源码目录下 device/rockchip/rv1126_rv1109/BoardConfig-alientek-rv1126.mk，此文件包含整个 SDK 包所需要的配置信息，文件内容如下所示：

示例代码 4.3.1 BoardConfig-alientek-rv1126.mk 文件内容

```
1  #!/bin/bash
2
3  # Target arch
4  export RK_ARCH=arm
5  # Uboot defconfig
6  export RK_UBOOT_DEFCONFIG=alientek_rv1126
```

```
# Uboot image format type: fit(flattened image tree)
export RK_UBOOT_FORMAT_TYPE=fit
# Kernel defconfig
export RK_KERNEL_DEFCONFIG=alientek_rv1126_defconfig
# Kernel defconfig fragment
export RK_KERNEL_DEFCONFIG_FRAGMENT=
# Kernel dts
export RK_KERNEL_DTS=rv1126-alientek
# boot image type
export RK_BOOT_IMG=zboot.img
# kernel image path
export RK_KERNEL_IMG=kernel/arch/arm/boot/zImage
# kernel image format type: fit(flattened image tree)
export RK_KERNEL_FIT_ITS=boot.its
# parameter for GPT table
export RK_PARAMETER=parameter-buildroot-fit.txt
# Buildroot config
export RK_CFG_BUILDROOT=alientek_rv1126
# Recovery config
export RK_CFG_RECOVERY=alientek_rv1126_recovery
# Recovery image format type: fit(flattened image tree)
export RK_RECOVERY_FIT_ITS=boot4recovery.its
# ramboot config
export RK_CFG_RAMBOOT=
# Pcba config
export RK_CFG_PCBA=
# Build jobs
export RK_JOBS=2
# target chip
export RK_TARGET_PRODUCT=rv1126_rv1109
# Set rootfs type, including ext2 ext4 squashfs
export RK_ROOTFS_TYPE=ext4
# rootfs image path
export RK_ROOTFS_IMG=rockdev/rootfs.${RK_ROOTFS_TYPE}
# Set ramboot image type
export RK_RAMBOOT_TYPE=
# Set oem partition type, including ext2 squashfs
export RK_OEM_FS_TYPE=ext2
# Set userdata partition type, including ext2, fat
export RK_USERDATA_FS_TYPE=ext2
#OEM config
export RK_OEM_DIR=oem_ipc
# OEM build on buildroot
```

```
50  export RK_OEM_BUILDIN_BUILDROOT=YES
51  #userdata config, if not define this, system will format by
RK_USERDATA_FS_TYPE
52  export RK_USERDATA_DIR=userdata_normal
53  #misc image
54  export RK_MISC=wipe_all-misc.img
55  #choose enable distro module
56  export RK_DISTRO_MODULE=
57  # Define pre-build script for this board
58  export RK_BOARD_PRE_BUILD_SCRIPT=app-build.sh
59  # Define package-file for update.img
60  export RK_PACKAGE_FILE=rv1126_rv1109-package-file
```

第 4 行：开发板的架构为 ARM。

第 6 行：Uboot 的配置文件为 alientek_rv1126_defconfig。

第 8 行：Uboot 的镜像格式为 fit。

第 10 行：kernel 的配置文件为 alientek_rv1126_defconfig。

第 14 行：kernel 的设备树为 rv1126-alientek.dts。

第 16 行：内核的镜像类型为 zboot.img（此文件包含 zImage、设备树和 logo）。

第 18 行：内核镜像文件为 zImage。

第 20 行：kernel 镜像的加载位置配置文件为 boot.its。

第 22 行：开发板的分区文件，emmc 会按照此分区文件进行分区。

第 24 行：buildroot 的默认配置文件为 alientek_rv1126_defconfig。

第 26 行：recovery 的默认配置文件为 alientek_rv1126_recovery_defconfig。

第 28 行：recovery 镜像的加载位置配置文件为 boot4recovery.its。

第 34 行：以 2 进程进行编译。

第 36 行：指定芯片为 rv1126_rv1109。

第 38 行，文件系统的格式为 ext4，只读文件系统可以设置为 spuashfs 格式。

第 40 行，文件系统的路径保存到 rockdev/rootfs.ext4。

第 44 行，oem 分区的格式类型为 ext2，只读 oem 可以设置为 spuashfs。

第 46 行，userdata 分区的格式类型为 ext2。

第 48 行，oem 的配置文件夹为 oem_ipc，在 device/rockchip/oem 目录下。

第 50 行，在 buildroot 上创建 oem。

第 52 行，userdata 的配置文件夹为 userdata_normal，在 device/rockchip/userdata/目录下。

第 54 行，misc 分区使用 img 文件为 wipe_all-misc.img。

第 58 行，模块编译的先后顺序，由 app-build.sh 文件控制。

第 60 行，打包的配置文件为 rv1126_rv1109-package-file。

从上面的配置文件看出来，RK 官方都是使用 XXXX.mk 文件，如果我们要基于原厂的 SDK 包进行开发，首先要先配置此文件。

## 4.4 全自动编译

**编译前必须看 3.1 小节，重点是第 51 页安装软件。**

第一次编译的时候需要下载很多压缩包，有些压缩包下载不了的，所以笔者把下载好的压

缩包打包成 dl.tar.gz，这样大家编译的时候不用下载了。此文件在**开发板光盘 A-基础资料→01、程序源码→02、buildroot 下载源码包→bl.tar.gz**，此压缩包拷贝到 Ubuntu 系统下，这边笔者拷贝到家目录。在源码目录下运行此命令进行创建 dl 目录：

```
mkdir buildroot/dl/ -p
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ mkdir buildroot/dl/ -p
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 4.4.1 创建 dl 目录

创建完成后，把 dl.tar.gz 解压到 buildroot/dl 目录下，如下命令所示：

```
tar -axvf ~/dl.tar.gz -C buildroot/dl/
```

运行结果如下图所示：

```
dl/weston-8.0.0.tar.xz
dl/wireless_tools.30.pre9.tar.gz
dl/wpa_supplicant-2.6.tar.gz
dl/x264-ba24899b0bf23345921da022f7a51e0c57dbe73d.tar.gz
dl/x265_2.5.tar.gz
dl/xbitmaps-1.1.2.tar.bz2
dl/xcb-proto-1.13.tar.bz2
dl/xcb-util-0.4.0.tar.bz2
dl/xcb-util-image-0.4.0.tar.bz2
dl/xcb-util-keysyms-0.4.0.tar.bz2
dl/xcb-util-renderutil-0.3.9.tar.bz2
dl/xcb-util-wm-0.4.1.tar.bz2
dl/xkbcomp-1.4.2.tar.bz2
dl/xkeyboard-config-2.23.1.tar.bz2
dl/XML-Parser-2.44.tar.gz
dl/xorgproto-2018.4.tar.bz2
dl/xserver_xorg-server-1.20.4_2019_12_16.tar.gz
dl/xtrans-1.4.0.tar.bz2
dl/xz-5.2.3.tar.bz2
dl/zbar-854a5d97059e395807091ac4d80c53f7968abb8f.tar.gz
dl/zeromq-4.1.6.tar.gz
dl/zip30.tgz
dl/zlib-1.2.11.tar.xz
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 4.4.2 解压 dl.tar.gz 文件

需要查看 buildroot/dl 目录下有没有很多压缩包，可以使用以下命令查看：

```
ls buildroot/dl/
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk1126$ ls buildroot/dl/
acl-2.2.52.src.tar.gz                                  lzip-1.19.tar.gz
alsa-lib-1.1.5.tar.bz2                                 lzo-2.10.tar.gz
alsa-plugins-1.1.5.tar.bz2                             m4-1.4.18.tar.xz
alsa-utils-1.1.5.tar.bz2                               Mako-1.1.2.tar.gz
android-tools_4.2.2+git20130218-3ubuntu41.debian.tar.gz  mediastreamer-2.14.0.tar.gz
android-tools_4.2.2+git20130218.orig.tar.xz            memtester-4.3.0.tar.gz
atk-2.36.0.tar.xz                                      mesa-20.3.4.tar.xz
attr-2.4.47.src.tar.gz                                 meson-0.54.2.tar.gz
autoconf-2.69.tar.xz                                   mkfontdir-1.0.7.tar.bz2
automake-1.15.1.tar.xz                                 mkfontscale-1.1.3.tar.bz2
avahi-0.7.tar.gz                                       mpdecimal-2.4.1.tar.gz
bash-4.4.12.tar.gz                                     mpg123-1.25.15.tar.bz2
bctoolbox-0.4.0.tar.gz                                 mtdev-1.1.4.tar.bz2
bdftopcf-1.1.tar.bz2                                   mysql-5.1.73.tar.gz
bison-3.0.4.tar.xz                                     ncurses-6.0.tar.gz
bluez-5.50.tar.xz                                      nginx-1.12.2.tar.gz
busybox-1.27.2.tar.bz2                                 nginx-http-flv-live-v1.2.8.tar.gz
bzip2-1.0.6.tar.gz                                     ninja-v1.8.2.tar.gz
cairo-1.16.0.tar.xz                                    NotoSansSC.zip
cantarell-fonts-0.0.25.tar.xz                          ntfs-3g_ntfsprogs-2017.3.23.tgz
cgicc-3.2.16.tar.bz2                                   ntp-4.2.8p10.tar.gz
clang-9.0.1.src.tar.xz                                 numpy-1.18.2.tar.gz
connman-1.35.tar.xz                                    opencv3-3.4.12.tar.gz
coreutils-8.30.tar.xz                                  opencv3-4.7.0.tar.gz
curl-7.75.0.tar.xz                                     opencv4-4.5.5.tar.gz
Cython-0.29.21.tar.gz                                  opencv-4.7.0.tar.gz
db-5.3.28.NC.tar.gz                                    openssl-1.1.1h.tar.gz
```

图 4.4.3 查看 dl 目录下的压缩包

注意：源码的压缩包都在“buildroot/dl”目录，不要放到其它目录。否则就会报下载出错，国内的环境有些源码下载不了。如果报一下错误，首先看源码目录是否放正确。

RK 官方提供了一按键编译功能，使用一条命令即可编译出镜像文件，在官方的源码目录下运行以下命令即可：

```
./build.sh lunch //选择 1
./build.sh
```

编译成功后运行结果如下图所示：

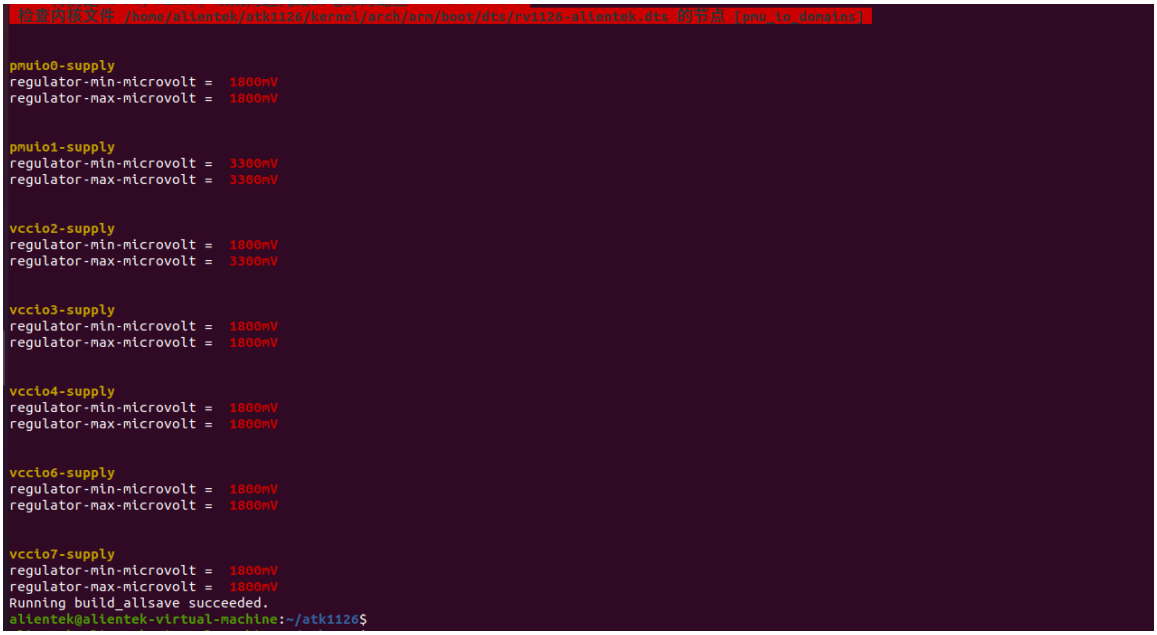



图 4.4.4 全自动编译

## 4.5 U-Boot 编译和配置

### 4.5.1 uboot 的编译

RK 官方提供了很方便的编译方法，直接运行以下./build.sh uboot 命令可以编译 uboot。运行命令如下所示：

```
./build.sh uboot
```

运行结果如下图所示：

```
  Data Size:    209938 Bytes = 205.02 KiB = 0.20 MiB
  Architecture: ARM
  Load Address: 0x08400000
  Hash algo:    sha256
  Hash value:   ea6f0c244022a9c57dfc2336567dacc4a54dde02046894c16e9c08cdc1605117
 Image 2 (fdt)
  Description:  U-Boot dtb
  Created:      Wed Oct 26 16:09:25 2022
  Type:         Flat Device Tree
  Compression:  uncompressed
  Data Size:    10931 Bytes = 10.67 KiB = 0.01 MiB
  Architecture: ARM
  Hash algo:    sha256
  Hash value:   a362d022dea92b84b2a8e3ea57aeaf9a33f94cf6bac545548731bf96db970c14
 Default Configuration: 'conf'
 Configuration 0 (conf)
  Description:  rv1126-evb
  Kernel:       unavailable
  Firmware:     optee
  FDT:          fdt
  Loadables:    uboot
/home/alientek/atk-rv1126/prebuilts/gcc/linux-x86/arm/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc
pack loader ok.(rv1126_spl_loader_v1.08.108.bin)(0.01)
pack loader okay! Input: /home/alientek/atk-rv1126/rkbin/RKBOOT/RV1126MINIALL.ini
/home/alientek/atk-rv1126/u-boot

Image(no-signed, version=0):  uboot.img (FIT with uboot, trust...) is ready
Image(no-signed):  rv1126_spl_loader_v1.08.108.bin (with spl, ddr, usbplug) is ready
pack uboot.img okay! Input: /home/alientek/atk-rv1126/rkbin/RKTRUST/RV1126TOS.ini

Platform RV1126 is build OK, with new .config(make rv1126_defconfig -j8)
Wed Oct 26 16:09:25 CST 2022
Running build_uboot succeeded.
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 4.5.1.1 编译 uboot 图

编译完成后，会在 u-boot 目录下生成 uboot.img 文件和 rv1126_spl_loader_v1.08.108.bin 文件，uboot.img 文件就是我们要烧录的 img 镜像，rv1126_spl_loader_v1.08.108.bin 是启动引导 uboot 的文件，就是在第 3.7.2 章节里的 MiniLoaderAll.bin 文件。查看结果如下所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126/u-boot$ ls
api     config.mk      drivers   fs        Licenses   post                        snapshot.commit  tee.digest  u-boot.bin          uboot.img          u-boot-nodtb.digest
arch    configs        dts       include   MAINTAINERS PREUPLOAD.cfg             spl              test        u-boot.cfg          u-boot.lds         u-boot.srec
board   disk           env       Kbuild    Makefile   README                      System.map       tools       u-boot.cfg.configs  u-boot.map         u-boot.sym
cmd     doc            examples  Kconfig   make.sh    rv1126_spl_loader_v1.08.108.bin tee.bin      tpl         u-boot.dtb          u-boot-nodtb.bin
common  Documentation  fit       lib       net        scripts                     tee.bin.gz       u-boot      u-boot-dtb.bin      u-boot-nodtb.bin.gz
alientek@alientek-virtual-machine:~/atk-rv1126/u-boot$
```

图 4.5.1.2 查看编译后的 Uboot 目录

### 4.5.2 uboot 和 SPL 文件单独烧录

把这两个文件拷贝到 Windows 下，打开 RKDevTool 工具，导入配置选项“ATK-DLRV1126 出厂系统配置.cfg”到软件里面，ATK-DLRV1126 开发板进入 LOADER 模式(不能进入 MASKROM)。在方框里面对“1”和“3”勾选上， Loader 分区配置 rv1126_spl_loader_v1.08.108.bin；Uboot 分区配置 uboot.img。配置结果如下图所示：

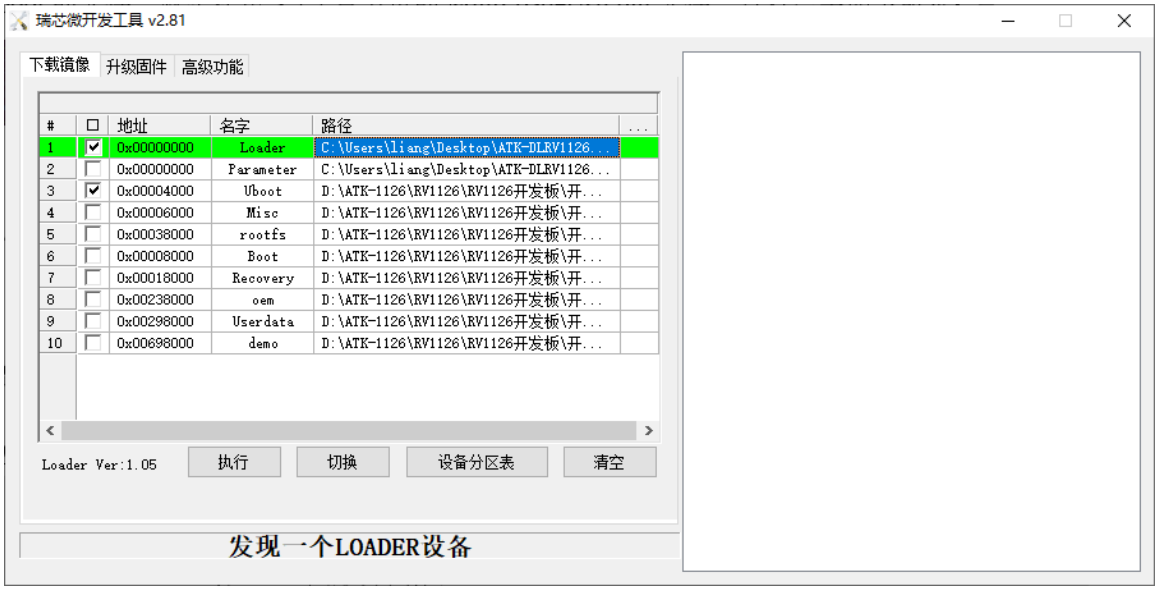



图 4.5.2.1 上位机单独烧录 uboot 图

点击执行即可烧录。烧录完成后系统会自动重启，看看打印信息就知道有没有编译成功了，如下图所示：

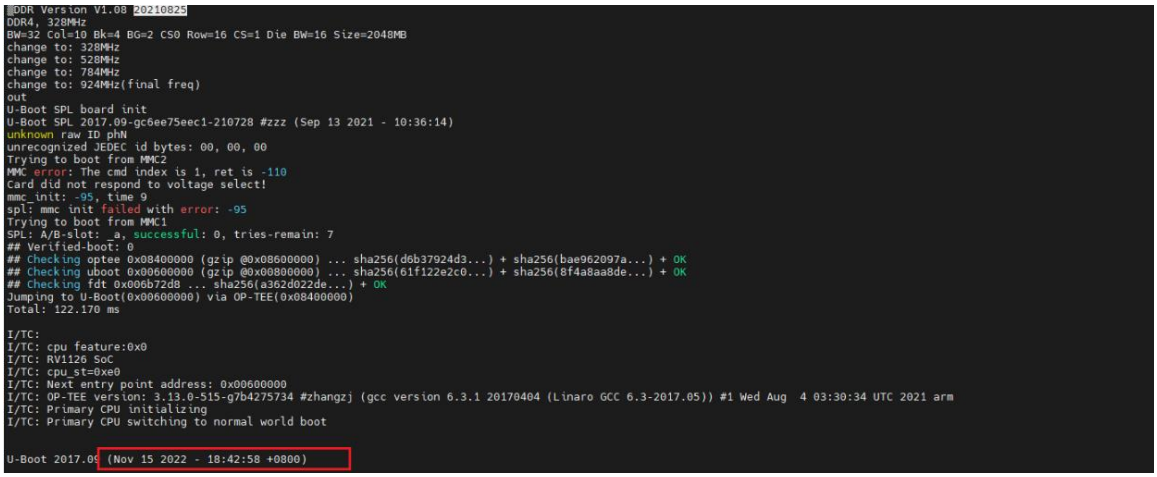



图 4.5.2.2 新烧录的 uboot 启动信息

图 4.5.2.2 中的红色框里打印日期为“2022-11-15 18:42:58”。

### 4.5.3 uboot 的配置

如果想要配置自己的 uboot，可以运行以下命令进行操作，在 SDK 源码目录下:

```
cd u-boot //跳转到 uboot 目录下
make alientek_rv1126_defconfig //选择要修改 Uboot 配置文件，从板级文件知道
make menuconfig //进入图形界面配置
```

运行结果如下所示：

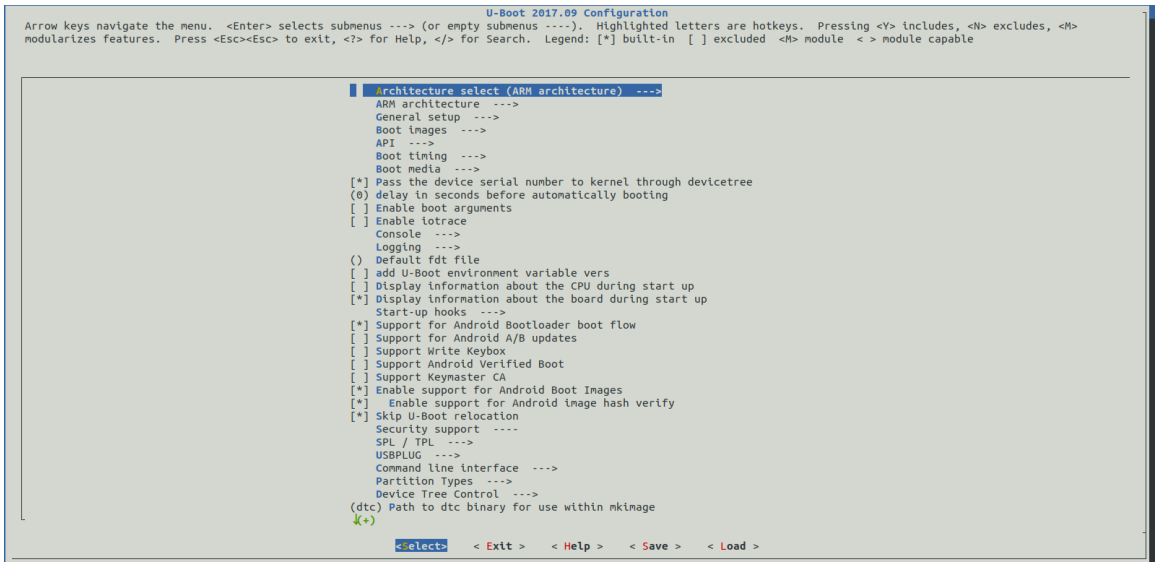



图 4.5.3 menuconfig 图形配置

从图 4.5.3 中可以看出 RK 官方是使用 2017.09 月份的 uboot 进行修改，我们就可以配置自己的 Uboot 了，配置完运行以下命令进行保存配置文件。

```
make savedefconfig //把.config 保存为 defconfig
cp defconfig configs/alientek_rv1126_defconfig // 更新修改好的配置文件到
```

```
alientek_rv1126_defconfig
    ./make.sh uboot //编译 uboot
```

配置完成后我们可以直接用 build.sh 脚本进行编译了，就会生成我们想要的 uboot.img 文件和 rv1126_spl_loader_v1.08.108.bin。可以把这几条命令写成脚本，就方便我们测试了。

## 4.6 kernel 编译和配置

### 4.6.1 kernel 的编译

编译 kernel 和 uboot 一样命令很简单，运行以下命令即可编译：

```
./build.sh kernel
```

运行此命令结果如下所示：

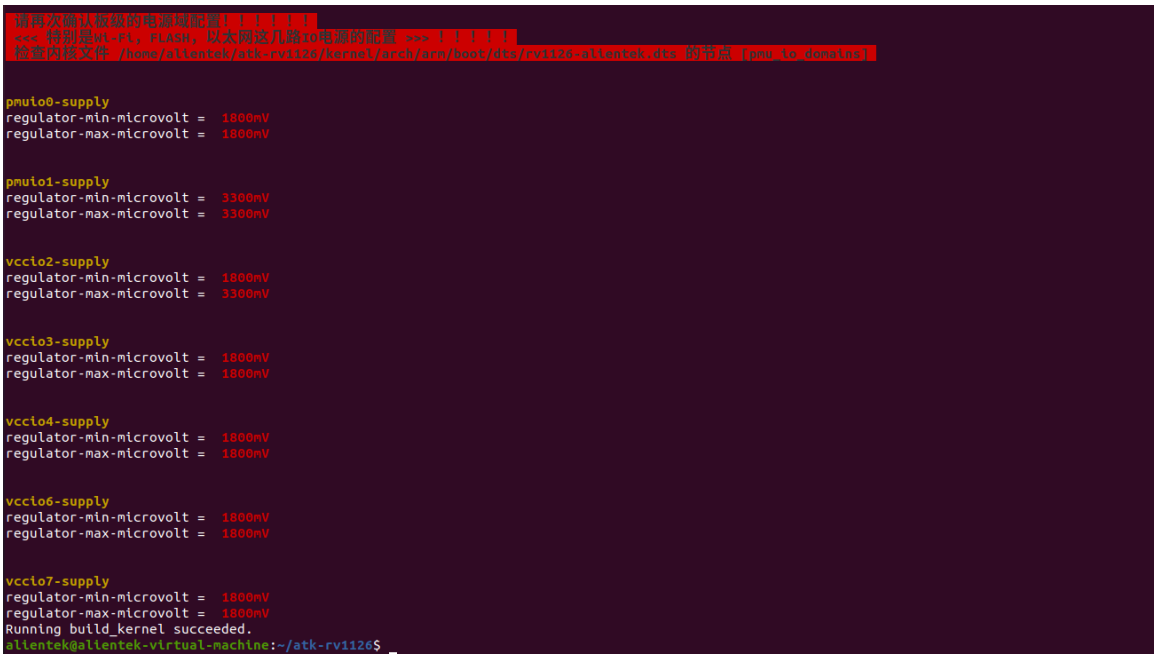



图 4.6.1.1 编译 kernel 图

```
655    &pmu_io_domains {
656        status = "okay";
657
658        pmuio0-supply = <&vcc1v8_pmu>;
659        pmuio1-supply = <&vcc3v3_sys>;
660        vccio2-supply = <&vccio_sd>;
661        vccio3-supply = <&vcc_1v8>;
662        vccio4-supply = <&vcc_1v8>;
663        vccio5-supply = <&vcc_3v3>;
664        vccio6-supply = <&vcc_1v8>;
665        vccio7-supply = <&vcc_1v8>;
666    };
```

图 4.6.1.2 设备树的电源域配置

图 4.6.2 中警告是 build.sh 打印出来的提醒我们注意设备树的电源域(图 4.6.1.2)和硬件要一

致，如果电压不一致可能会烧 CPU，千万、千万、千万不要修改电源域的配置。

编译完成后在“kernel”目录下生成 zboot.img(配置文件里面决定生成为 zboot.img)。可以跳转到内核里面，运行“ls”命令查看，结果如下所示：


图 4.6.1.3 kernel 文件夹目录

### 4.6.2 kernel 的烧录

和单独烧录 uboot 系统一样，先把 zboot.img 文件拷贝到 Windows 系统里面，ATK-DLRV1126 开发板进入 LOADER 模式，打开烧录软件，在方框里面对“6”勾选上，Boot 分区配置 zboot.img。配置结果如下图所示：

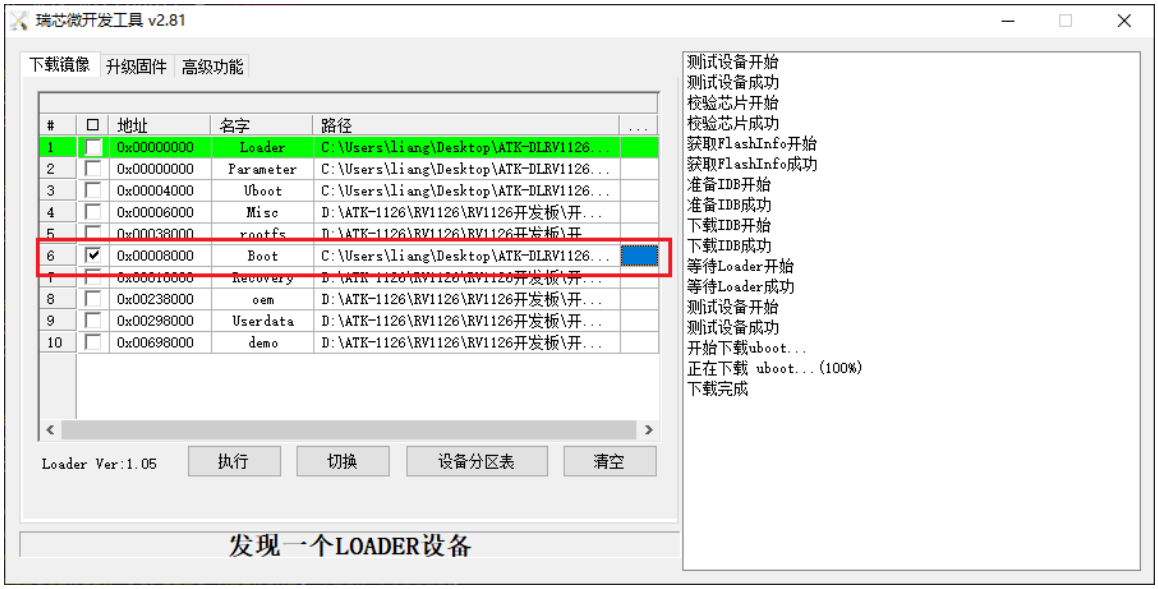



图 4.6.2.1 上位机烧录 kernel 配置图

点击运行即可进行烧录。烧录成功会自动重启，我们查看打印信息内核有没有替换成功，如下图所示：

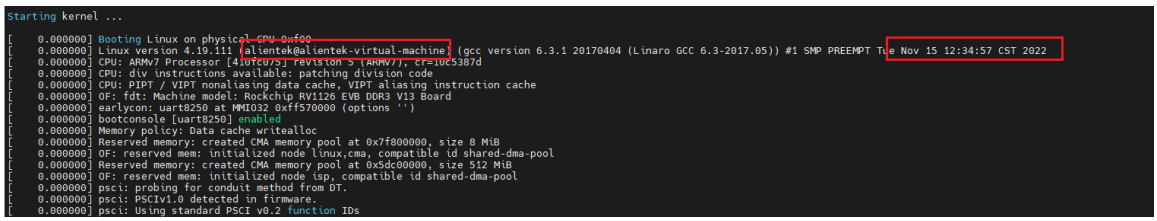



图 4.6.2.2 内核烧录成功打印信息

### 4.6.3 kernel 的配置

如果想修改内核，也可以运行以下命令进行修改：

```
cd kernel/ //跳转到 kernel
make ARCH=arm alientek_rv1126_defconfig //选择要修改的 kernel 配置文件
```

```
make ARCH=arm menuconfig //进入图形界面配置
```

运行结果如下所示：

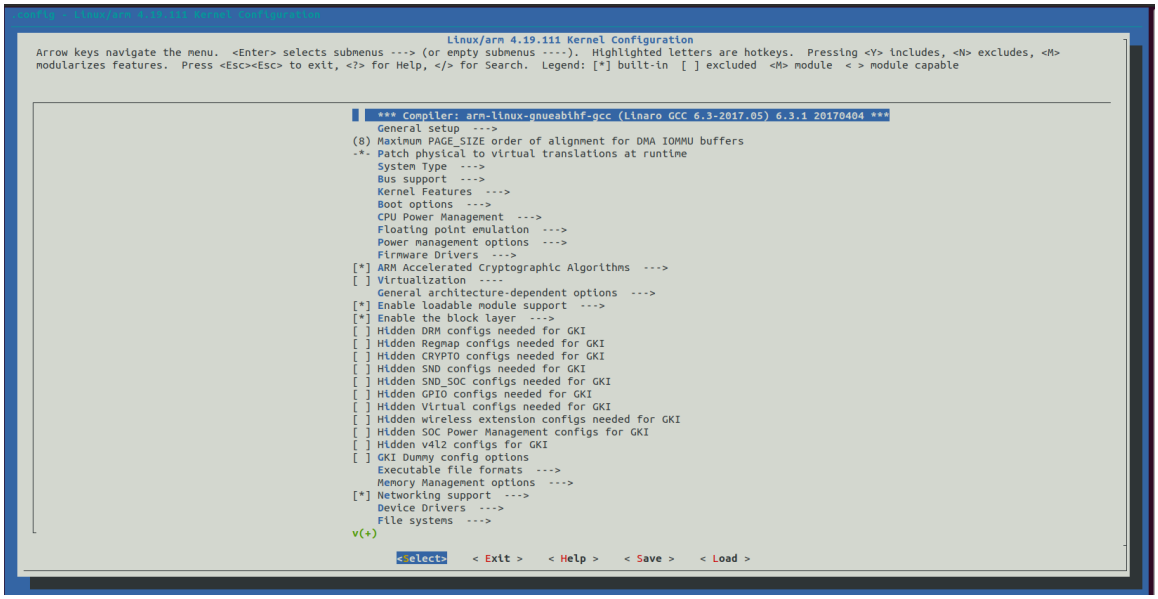



图 4.6.3.1 kernel menuconfig 配置图形界面

从图 4.6.3.1 中可以看出 RK 官方是使用 4.19.111 版本的 kernel 进行修改，我们就可以配置自己的 kernel 了，配置完运行以下命令进行保存配置文件。

```
make ARCH=arm savedefconfig  //把.config 保存为 defconfig
cp defconfig arch/arm/configs/alientek_rv1126_defconfig // 更新修改好的配置文件到 alientek_rv1126_defconfig
cd .. //跳转到源码目录
./build.sh kernel //编译内核
```

配置完成后我们可以直接用 build.sh kernel 脚本进行编译了，就会生成我们想要的 zboot.img 文件。可以把这几条命令写成脚本，就方便我们测试了。

### 4.6.4 logo 的修改

在源码目录下，进入内核目录，运行以下命令：

```
ls
```

运行结果如下所示：


图 4.6.4.1 查看 kernel 源码目录内容

在图 4.6.4.1 中红色框住的两个 bmp 文件就是启动的 logo。logo.bmp 为 uboot 的 logo；logo_kernel.bmp 为内核的 logo。如果要替换自己的 logo 替换这两个文件即可。然后重新编译内核。注意：两个文件的 8 位的 bmp 格式的图片。

### 4.6.5 单个设备树编译

出厂系统是把多个设备树和内核编译成一个 boot.img 文件。通过 ADC 读取屏幕不同的 ADC 值来匹配对应的设备树。当我们只需要一个设备的时候如何编译。首先在源码目录下打开 build.sh 文件，修改第六行，DEVICE_LCD(注意默认出厂值为 all，把正点原子所有 MIPI 屏设备树打包在一起)的值即可。如果购买的屏幕为 5.5 寸屏 720P 修改对应的值为 720。如下图所示：


图 4.6.5.1 5.5 寸屏 720P 的修改方法

购买屏幕为 10 寸屏 800P 修改对应的值为 800。如下图所示：


图 4.6.5.2 10 寸屏的修改方法

购买屏幕为 5.5 寸屏 1080P 修改对应的值为 1080。如下图所示：


图 4.6.5.3 5.5 寸屏 1080P 的修改方法

如果像出厂系统一样，兼容多个设备树，修改对应的值为 all，如下图所示：


图 4.6.5.4 把所有设备树打包

内核的脚本编译(在 build.sh 文件里)，如下图所示：

```
500 function build_kernel(){
501     check_config RK_KERNEL_DTS RK_KERNEL_DEFCONFIG || return 0
502
503     echo "============Start building kernel============"
504     echo "TARGET_ARCH          =$RK_ARCH"
505     echo "TARGET_KERNEL_CONFIG =$RK_KERNEL_DEFCONFIG"
506     echo "TARGET_KERNEL_DTS    =$RK_KERNEL_DTS"
507     echo "TARGET_KERNEL_CONFIG_FRAGMENT =$RK_KERNEL_DEFCONFIG_FRAGMENT"
508     echo "============================================="
509
510     cd kernel
511     make ARCH=$RK_ARCH $RK_KERNEL_DEFCONFIG $RK_KERNEL_DEFCONFIG_FRAGMENT
512     make ARCH=arm rv1126-alientek-1080p.dtb rv1126-alientek-800p.dtb
513     make ARCH=$RK_ARCH $RK_KERNEL_DTS.img -j$RK_JOBS
514     rm zboot.img
515     case "$DEVICE_LCD" in
516         "720")
517             ./scripts/mkmultidtb.py ATK-RV1126-EVB_720
518             ;;
519         "800")
520             ./scripts/mkmultidtb.py ATK-RV1126-EVB_800
521             ;;
522         "1080")
523             ./scripts/mkmultidtb.py ATK-RV1126-EVB_1080
524             ;;
525         *)
526             ./scripts/mkmultidtb.py ATK-RV1126-EVB
527             ;;
528     esac
529     ./scripts/mkbootimg  --kernel ./arch/arm/boot/zImage  --second resource.img -o zboot.img
530     if [ -f "$TOP_DIR/device/rockchip/$RK_TARGET_PRODUCT/$RK_KERNEL_FIT_ITS" ]; then
531         $COMMON_DIR/mk-fitimage.sh $TOP_DIR/kernel/$RK_BOOT_IMG \
532             $TOP_DIR/device/rockchip/$RK_TARGET_PRODUCT/$RK_KERNEL_FIT_ITS
533     fi
```

图 4.6.5.5 build_lernel 函数

上图可以看出，先编译内核和所有的设备树，然后调用 mkmultidtb.py 文件把对应的 zImage 和设备树重新打包 zboot.img。

## 4.7 recovery 编译和配置

Recovery 不是必须要的功能，板级配置了此功能，所以我们也需要编译。

### 4.7.1 recovery 编译

编译 recovery 的命令也是很简单的，命令如下图所示：

```
./build.sh recovery
```

运行此命令结果如下所示：

```
Image 2 (ramdisk)
 Description:  unavailable
 Created:      Thu Oct 27 12:34:32 2022
 Type:         RAMDisk Image
 Compression:  uncompressed
 Data Size:    6947975 Bytes = 6785.13 KiB = 6.63 MiB
 Architecture: ARM
 OS:           Linux
 Load Address: 0xffffff02
 Entry Point:  unavailable
 Hash algo:    sha256
 Hash value:   9db90c3918d65ebdb96c2cb36bf12268bd96c6292cf3ca90b44e36230605cfba
Image 3 (resource)
 Description:  unavailable
 Created:      Thu Oct 27 12:34:32 2022
 Type:         Multi-File Image
 Compression:  uncompressed
 Data Size:    341504 Bytes = 333.50 KiB = 0.33 MiB
 Hash algo:    sha256
 Hash value:   d7866d3e6f255b2e6cf0a7d3ac06bb209460bcc9f2cb20ea200b12d0bbec6422
Default Configuration: 'conf'
Configuration 0 (conf)
 Description:  unavailable
 Kernel:       kernel
 Init Ramdisk: ramdisk
 FDT:          fdt
one.
ou take 8:02.59 to build recovery
unning build_recovery succeeded.
lientek@alientek-virtual-machine:~/atk-rv1126$
```

图 4.7.1.1 编译 recovery 图

如果编译不过通常是网络问题，这边笔者把要下载的软件包放到：开发板光盘 A-基础资料→01、程序源码→02、buildroot 源码包→dl.tar.bz2，把此文件解压到 buildroot/dl 目录(如果没有此目录可以自行创建)。编译完成就会生成 buildroot/output/alientek_rv1126_recovery/images/recovery.img 文件。运行以下命令可以查看文件有么有生成：

```
ls buildroot/output/alientek_rv1126_recovery/images/recovery.img
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ ls buildroot/output/alientek_rv1126_recovery/images/recovery.img
buildroot/output/alientek_rv1126_recovery/images/recovery.img
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 4.7.1.1 查看 recovery.img 文件的生成

### 4.7.2 recovery 的烧录

先把 recovery.img 文件拷贝到 Windows 系统里面，ATK-DLRV1126 开发板进入 LOADER 模式，打开烧录软件，在方框里面对“7”勾选上，Recovery 分区配置 recovery.img。配置如下图所示：

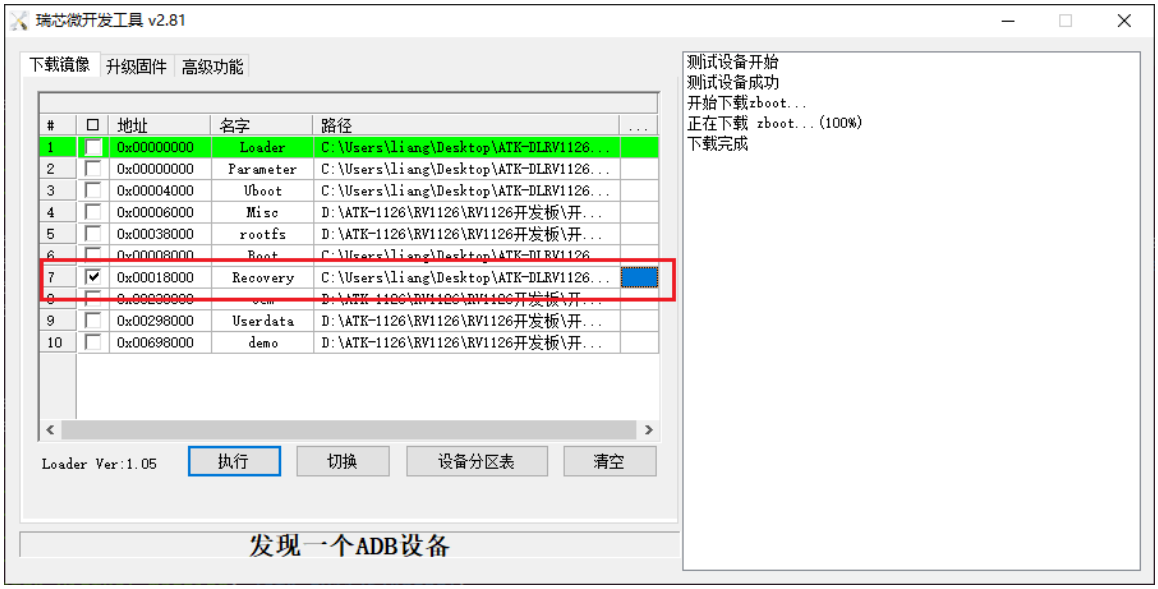



图 4.7.2.1 recovery 烧录配置

### 4.7.3 recovery 的配置

在 SDK 包的源码目录下，运行以下命令即可配置 recovery：

```
source envsetup.sh alientek_rv1126_recovery //配置 recovery 的板级信息
make menuconfig //进入图形界面配置
```

运行结果如下图所示：

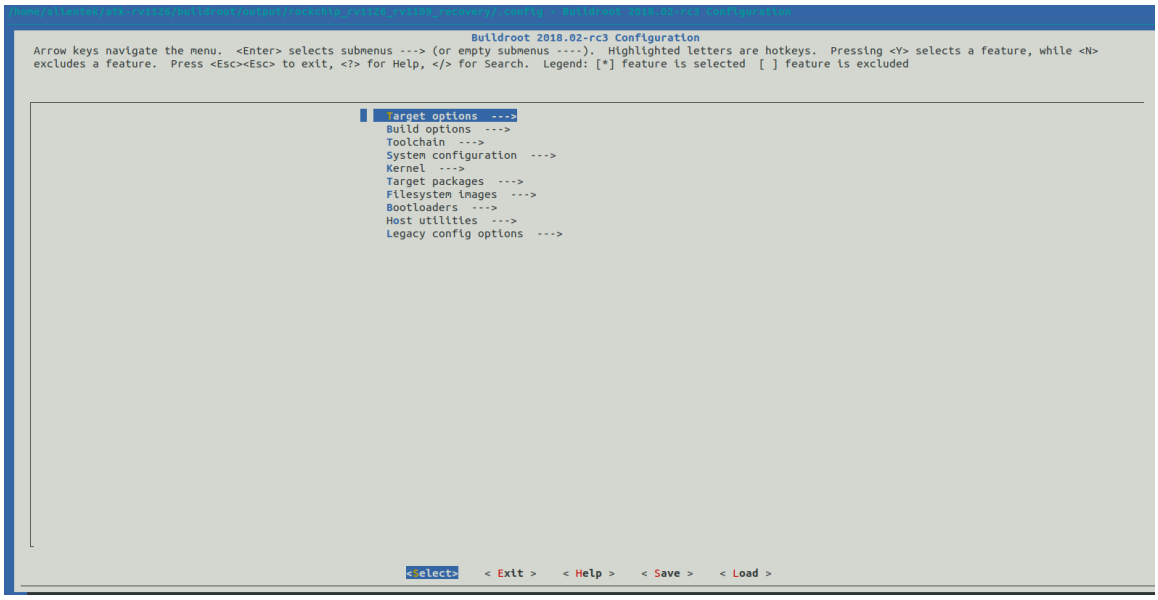



图 4.7.3.1 配置 recovery 图

从图中可以看出 recovery 也是一个文件系统，只是此系统 RK 官方做了修改，是用 buildroot 2018.02 做的。配置完成后我们可以运行以下代码就行保存：

```
make savedefconfig //保存到 buildroot/configs/alientek_rv1126_recovery_defconfig 里
./build.sh recovery //重新编译 recovery
```

保存成功后，可以直接运行./build.sh recovery 进行编译。

## 4.8 rootfs 编译和配置

### 4.8.1 rootfs 的编译

编译文件系统，还是一条命令解决，命令如下图所示：

```
./build.sh rootfs
```

编译结果如下所示：

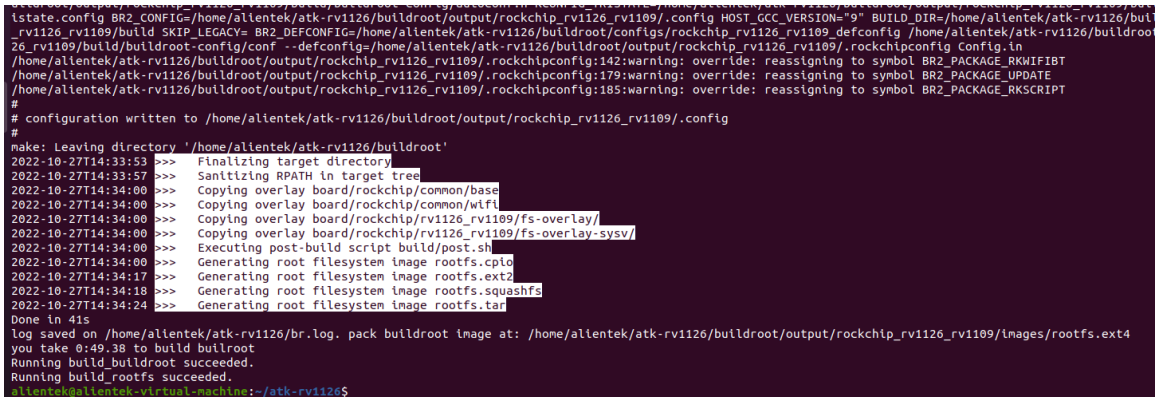



图 4.8.1.1 编译 rootfs 图

编译完成就会在源码目录下创建一个“rockdev”文件夹，跳转到此目录下查看究竟生成了那些文件，查看结果如下图所示：


图 4.8.1.2 查看 rockdev 目录

上图可以看出生成了“demo.img”、“oem.img”和“rootfs.ext4”，rootfs.ext4 其实就是文件系统 rootfs.img。rootfs.ext4 文件就是一个软连接“../buildroot/output/rockchip_rv1126_rv1109/images/rootfs.ext2”文件。

### 4.8.2 rootfs 的烧录

先把 rootfs.ext4、oem.img 和 demo.img 文件拷贝到 Windows 系统里面，ATK-DLRV1126 开发板进入 LOADER 模式，打开烧录软件，在方框里面对“5”、“8”和“10”勾选上，在配置我们要烧录的文件，rootfs 分区配置 rootfs.ext4;oem 分区配置 oem.img;demo 分区配置 demo.img。配置如下图所示：

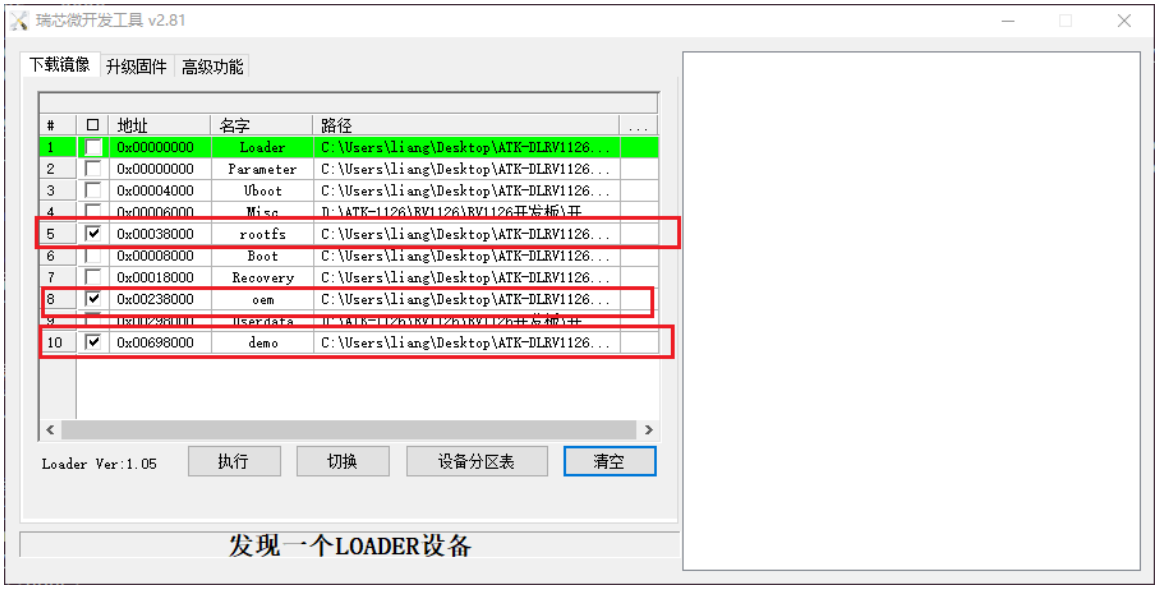



图 4.7.2.1 文件系统相关的配置

点击执行即可烧录 rootfs、oem 和 demo。

### 4.8.3 rootfs 的配置

- buildroot 配置

在 SDK 包源码目录下，运行以下命令进行配置 buildroot：

```
source envsetup.sh alientek_rv1126 //配置 buildroot 对应 defconfig
make menuconfig //进入图形化界面
```

运行结果如下所示：

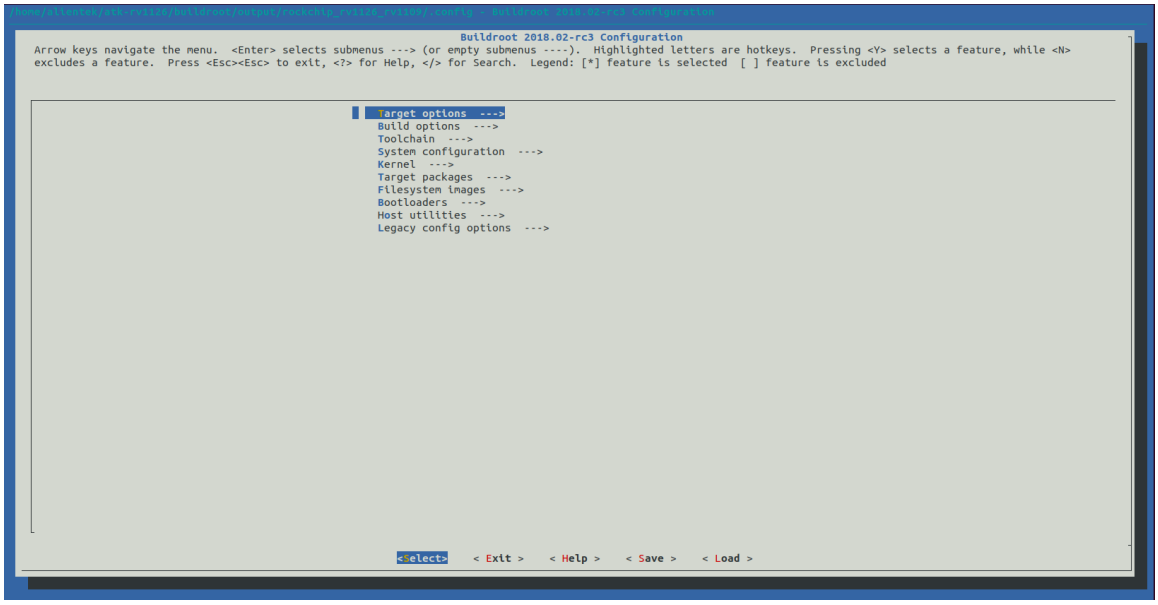



图 4.8.3.1 buildroot 配置图

配置完成好保存。运行以下命令进行保存和重新编译(注意：buildroot 有时候不能生成一些配置选项，要运行./build.sh cleanall 命令清除，在编译)

```
make savedefconfig //保存配置文件到 buildroot/configs/alientek_rv1126_defconfig
```

```
./build.sh rootfs
```

- busybox 配置

  在 SDK 包源码目录下，运行以下命令进行配置 busybox：

```
source envsetup.sh alientek_rv1126 //配置 buildroot 对应 defconfig
make busybox-menuconfig //进入图形化界面
```

运行结果如下所示：

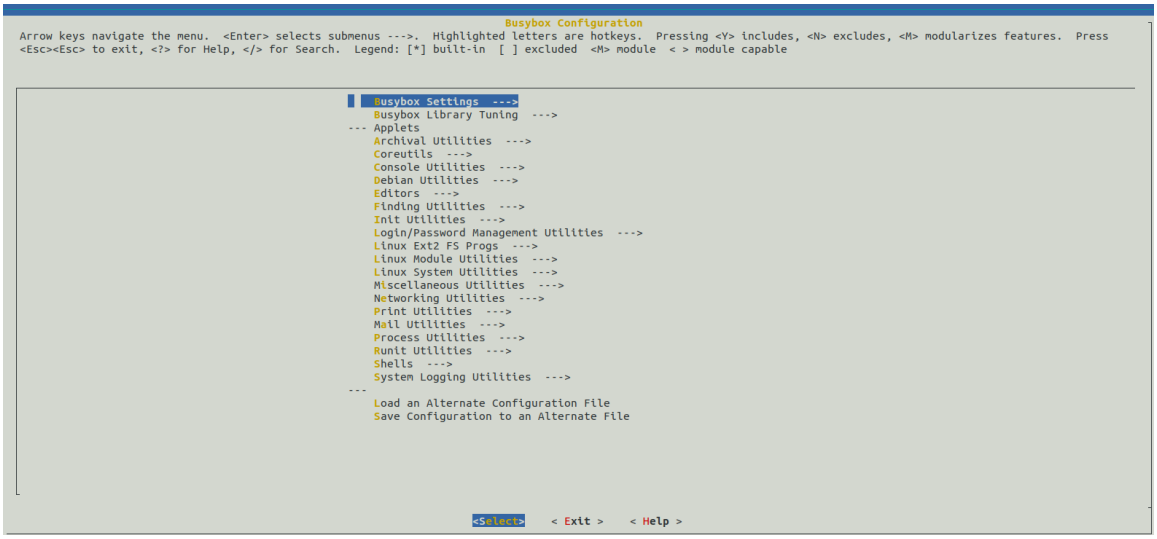



图 4.8.3.2 busybox 配置图

配置完成后，放回最上层，如图 4.8.3.2 中所示，选择“save Configuration to an Alternate File”进行保存到“.config”文件。运行以下命令进行保存刚刚修改的配置。

```
make busybox-update-config
```

将修改保存到配置文件 board/rockchip/common/base/busybox.config。重新编译文件系统即可。

## 4.9 编译第三方库或者 APP

整个 SDK 包里面包含了很多第三方库和 APP，如果想单独编译这些可以运行以下命令即可，运行以下命令即可：

```
./build.sh external/mpp app/mediaserver
./build.sh rootfs
```

编译完成后可以重新烧录 rootfs.img。

## 4.10 编译 BSP 包

我们可以编译出只包含音视频编解码库、NPU 库以及头文件。注意：BSP 包不包含文件系统的。

```
source envsetup.sh alientek_rv1126_libs
make -j12
```

编译 BSP 生成的目录 buildroot/output/alientek_rv1126_libs/BSP。

## 4.11 固件打包

在上几章节中我们生成了很多的 img 文件，每次都要在不同的目录下拷贝很麻烦，官方提

供了一个脚本，运行脚本命令如下所示(在 SDK 源码目录下运行)：

```
source envsetup.sh alientek_rv1126 //选择环境变量 alientek_rv1126
./mkfirmware.sh
```

运行结果如下图所示：

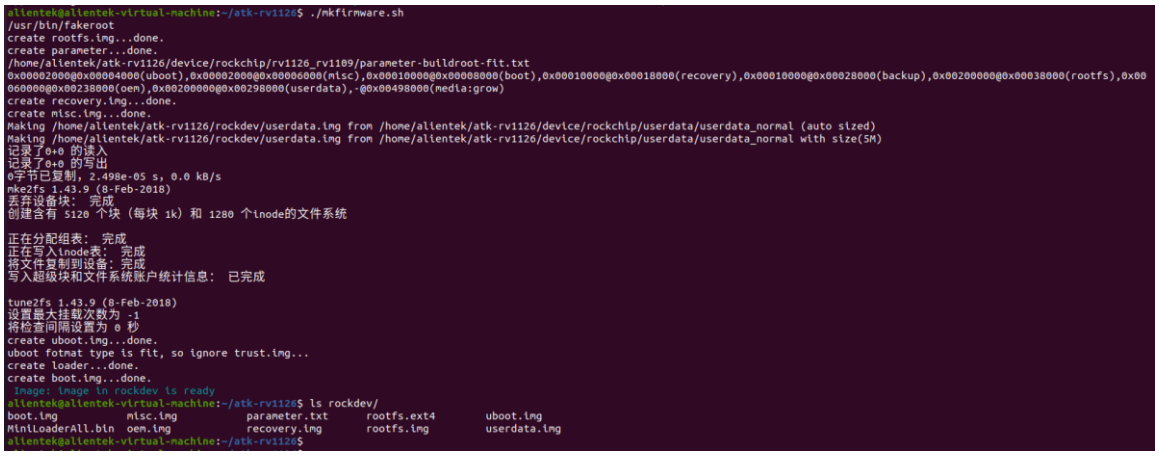



图 4.11.1 固件打包

打包成功会在源码目录下新建一个“rockdev”目录，把需要烧录的文件打包进里面。我们可以把这些文件通过 FTP 拷贝到对应的 Windows 目录下进行烧录。也可以使用 rkflash.sh 脚本在 Ubuntu 进行烧录。到这里 SDK 已经编译完成了，我们就跳转到“rockdev”目录下查看生成了那些文件，运行“ls -l”查看文件，如下图所示：


图 4.11.2 rockdev 目录文件

可以看出很多文件都是通过软连接，简单的讲解每个文件的作用：

boot.img：里面包含了设备树、kernel 和 logo。注：里面有多个设备树文件。

demo.img：正点原子官方出厂 demo。

MiniLoaderAll.bin：此文件负责初始化 DDR，引导 Uboot。相当于 SPL 或者 TF-A。

misc.img：常用来作为系统升级时或者恢复出厂设置时使用。

oem.img：提供给厂商用的分区文件，像 RK 把他们自己库都放到这里。

parameter.txt：分区相关的文件。

recovery.img：升级相关的文件系统。

rootfs.img：buildroot 的文件系统。

uboot.img：里面包含 uboot 和设备树。

update.img：整个 SDK 的 img。

userdata.img：用户分区。

# 第五章 RKMedia 编译和使用

## 5.1 RKMedia 编译

Rkmedia 是 RK 官方封装一层简易的 API，把 RGA、MPP、RKNN 等等这些接口封装成高级的接口。在 SDK 官方的源码目录下，运行以下命令进行跳转：

```
cd external/rkmedia/examples/
ls
```

运行命令结果如下所示：

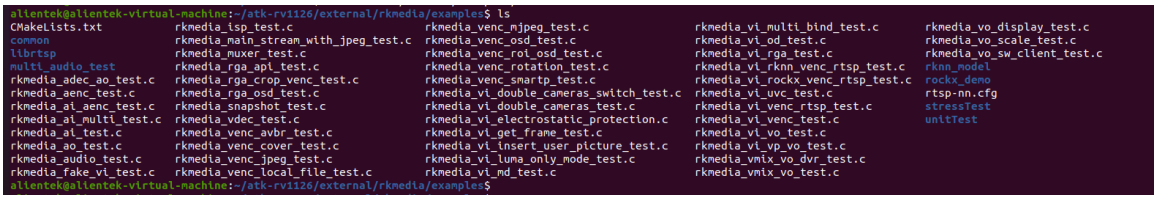



图 4.12.1.1 rkmedia 官方的 demo

里面有很多 C 文件的代码，可以结合 Rockchip_Developer_Guide_Linux_RKMedia_CN.pdf 文档(路径为：开发板光盘 A-基础资料→08、RV1126 参考资料→RV1126_RV1109→Multimedia →Rockchip_Developer_Guide_Linux_RKMedia_CN.pdf)进行修改和测试。编译方法如下命令所示：

```
source envsetup.sh alientek_rv1126 //选择环境变量 alientek_rv1126
make rkmedia-dirclean  //清除刚刚编译的 rkmedia
make rkmedia //重新编译 rkmedia
./build.sh rootfs //打包到文件系统里面
```

编译完成后重新烧录 oem.img 分区即可，因为 RK 官方把音频库都放在 oem.img 分区里面。

## 5.2 RKMedia 使用

在上一小节里面我们已经编译和烧录进开发板。接着就接上串口，开发板上电，进入终端。在终端里面输入“rkmedia_”在按 TAB 键，会显示很多命令，运行结果如下图所示：

```
[root@ATK-DLRV1126:/]# rkmedia_
rkmedia_adec_ao_test                    rkmedia_venc_roi_osd_test
rkmedia_aenc_test                       rkmedia_venc_rotation_test
rkmedia_ai_aenc_test                    rkmedia_venc_smartp_test
rkmedia_ai_test                         rkmedia_vi_double_cameras_switch_test
rkmedia_ao_test                         rkmedia_vi_double_cameras_test
rkmedia_audio_test                      rkmedia_vi_electrostatic_protection
rkmedia_face_capture_test               rkmedia_vi_get_frame_test
rkmedia_fake_vi_test                    rkmedia_vi_insert_user_picture_test
rkmedia_isp_test                        rkmedia_vi_luma_only_mode_test
rkmedia_main_stream_with_jpeg_test      rkmedia_vi_md_test
rkmedia_multi_ai_test                   rkmedia_vi_multi_bind_test
rkmedia_multi_ao_test                   rkmedia_vi_od_test
rkmedia_muxer_test                      rkmedia_vi_rga_test
rkmedia_rga_api_test                    rkmedia_vi_rknn_venc_rtsp_test
rkmedia_rga_crop_venc_test              rkmedia_vi_rockx_venc_rtsp_test
rkmedia_rga_osd_test                    rkmedia_vi_uvc_test
rkmedia_rockx_person_detect_test        rkmedia_vi_venc_rtsp_test
rkmedia_snapshot_test                   rkmedia_vi_venc_test
rkmedia_vdec_test                       rkmedia_vi_vo_test
rkmedia_venc_avbr_test                  rkmedia_vi_vp_vo_test
rkmedia_venc_cover_test                 rkmedia_vmix_vo_dvr_test
rkmedia_venc_jpeg_test                  rkmedia_vmix_vo_test
rkmedia_venc_local_file_test            rkmedia_vo_display_test
rkmedia_venc_mjpeg_test                 rkmedia_vo_scale_test
rkmedia_venc_osd_test
```

图 4.12.2.1 rkmedia 命令例程

这里列一些常用的例程(例程太多了和有些例程有 BUG)讲解如何使用这些命令：

### 5.2.1 rkmedia_ai_test

调用录音功能，录制 PCM 格式的音频文件。下表是使用的参数：

| 选项 | 描述 | 默认值 |
|---|---|---|
| -d | 音频输入节点名 | default |
| -r | 输出文件的采样率 | 16000 |
| -c | 输出文件的声道 | 2 |
| -o | 输出文件路径 | /tmp/ai.pcm |
| -s | 输出文件的帧数 | 1024 |
| -h | 查看帮助 | 无 |

使用方法，运行以下命令：

```
rkmedia_ai_test
amixer cset name='Digital Capture Volume' 120,120 //录音质量不行，可以设置录音的音量大小
```

运行的时候可以靠近对着 MIC 说好，这样就能录音了。运行结果如下图所示：

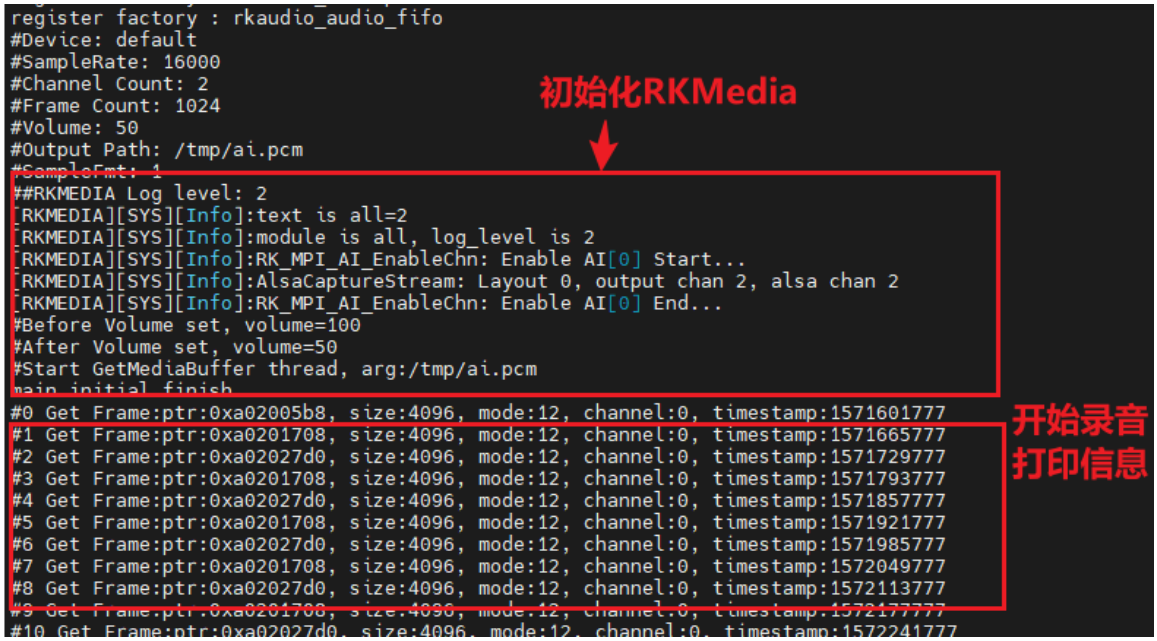



图 5.2.1.1 rkmedia_ai_test 运行打印信息

如果想结束程序，在终端里面按“Ctrl+C”结束程序。查看/tmp/ai.pcm 文件有么有数据产生，结果如下图所示：

```
[root@ATK-DLRV1126:/]# ls /tmp/ai.pcm  -l
-rw-r--r-- 1 root root 12894208 Nov 17 12:07 /tmp/ai.pcm
[root@ATK-DLRV1126:/]#
```

图 5.2.1.2 ai.pcm 的文件查看

### 5.2.2rkmedia_ao_test

播放 PCM 格式的音频文件，下表是使用的参数：

| 选项 | 描述 | 默认值 |
|---|---|---|
| -d | 音频输入节点名 | default |
| -r | 输入文件的采样率 | 16000 |
| -c | 输入的声道 | 2 |
| -i | 输入文件路径 | 无 |
| -s | 输入文件的帧数 | 1024 |
| -h | 查看帮助 | 无 |

使用方法：

```
rkmedia_ao_test -i /tmp/ai.pcm
```

就会播放我们刚刚录的声音了，运行结果如下所示：

```
#SampleFmt: 1
##RKMEDIA Log level: 2
[RKMEDIA][SYS][Info]:text is all=2
[RKMEDIA][SYS][Info]:module is all, log_level is 2
[RKMEDIA][SYS][Info]:RK_MPI_AO_EnableChn: Enable AO[0] Start...
[RKMEDIA][SYS][Info]:RK_MPI_AO_EnableChn: Enable AO[0] End...
#Before Volume set, volume=100
#After Volume set, volume=100
main initial finish
# TimeVal:64000us, ReadSize:4096
# TimeVal:64000us, ReadSize:4096
# TimeVal:64000us, ReadSize:4096
# TimeVal:64000us, ReadSize:4096
# TimeVal:64000us, ReadSize:4096
```

图 5.2.2.1 rkmedia_ao_test 播放 PCM 格式的音频

### 5.2.3 rkmedia_ai_aenc_test

调用录音功能，把输入的音频进行编码支持 MP3 、MP2、g711u、g771a 和 g726(测试发现 MP3 格式是不能录制的，跟代码发现在调用 rkmedia 库就报错了，估计是 SDK 版本问题)。下表是使用的参数：

| 选项 | 描述 | 默认值 |
| --- | --- | --- |
| -d | 音频输入节点名 | default |
| -r | 输出文件的采样率 | 16000 |
| -c | 输出文件的声道 | 2 |
| -o | 输出文件路径 | /tmp/aenc.mp2 |
| -l | 输出文件的帧数 | 1024 |
| -f | 输出文件的位深度 | s16 |
| -b | 输出文件的比特率 | 64000 |
| -t | 输出文件的编码类型 | MP2 |
| -h | 查看帮助 | 无 |

使用方法为：

```
rkmedia_ai_aenc_test -l 1152 -o aenc.mp2
```

对着 MIC 说话即可录音，如果想结束程序，在终端里面按“Ctrl+C”结束程序。运行结果如下图所示：

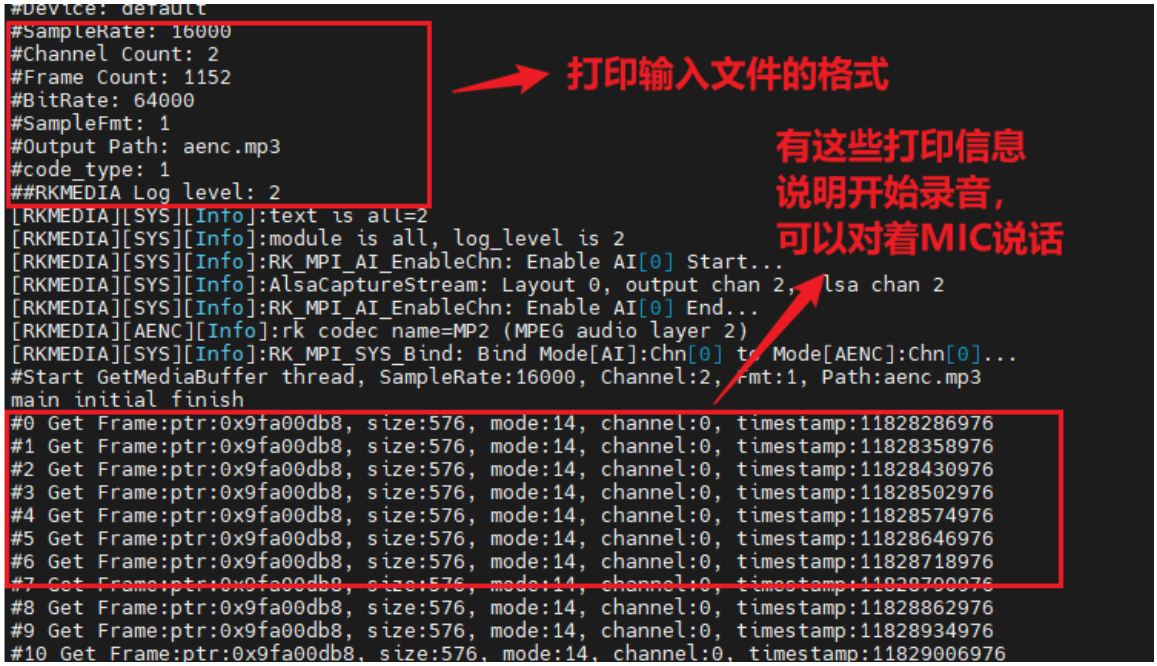



图 5.2.3.1 rkmedia_ai_aenc_test 录音 MP3 格式

在终端里面按“Ctrl+C”结束程序，通过 ADB 命令把“aenc.mp2”文件拷贝到 Windows 系统，点播放即可。

### 5.2.4 rkmedia_adec_ao_test

解码音频文件进行播放，实际测试只支持 g711u 和 g771a，没有办法解码 MP2 和 MP3 格式，不知道是不是 SDK 包的版本太新，以前版本还支持 AAC 现在已经不支持，查看源码发现，这些 MP3、MP2 解码结构体都没有定义。

```
typedef struct rkADEC_ATTR_MP3_S {
  // reserved
} ADEC_ATTR_MP3_S;

typedef struct rkADEC_ATTR_MP2_S {
  // reserved
} ADEC_ATTR_MP2_S;

typedef struct rkADEC_ATTR_G711A_S {
  RK_U32 u32Channels;
  RK_U32 u32SampleRate;
} ADEC_ATTR_G711A_S;

typedef struct rkADEC_ATTR_G711U_S {
  RK_U32 u32Channels;
  RK_U32 u32SampleRate;
} ADEC_ATTR_G711U_S;

typedef struct rkADEC_ATTR_G726_S {
  // reserved
} ADEC_ATTR_G726_S;
```

图 5.2.4.1 rkmedia_adec.h 头文件

下表是此命令的参数：

| 选项 | 描述 | 默认值 |
|---|---|---|
| -d | 音频输出节点名 | default |
| -r | 输入文件的采样率 | 16000 |
| -c | 输入文件的声道 | 2 |
| -o | 输入文件路径 | /tmp/aenc.mp3 |
| -l | 输入文件的帧数 | 1024 |
| -f | 输入文件的位深度 | s16 |
| -b | 输入文件的比特率 | 64000 |
| -t | 输入文件的编码类型 | mp3 |
| -h | 查看帮助 | 无 |

使用方法如下命令：

```
rkmedia_adec_ao_test -i aenc.g711u -t 3
```

“-t 3”指定解码格式为 g711u。刚刚已经说了实际测试只支持 g711u 或 g711a 格式，没有这两种格式的文件，可以使用 rkmedia_ai_aenc_test 命令进行录制。运行结果如下所示：


图 5.2.4.2 adec 音频解码打印信息

### 5.2.5 rkmedia_vi_get_frame_test

使用此命令前要先退出 QT 界面(点击设置→退出)，获取摄像头的数据(VI 表示摄像头的输入)，保存为 NV12 格式的视频格式，下表是此命令的参数：

| 选项 | 描述 | 默认值 | 备注 |
|---|---|---|---|
| -a/--aiq | 启动 ISP 功能，如果测试的摄像头为 MIPI 摄像头基本都需要开启 ISP 功能，参数为 aiq 文件所在文件夹路径。 | /oem/etc/iqfiles/ | 无 |
| -M | ISP 下的 multictx 开关 | 0 | 0：表示关闭<br>1：开启 |
| -I | ISP 下的摄像头切换 | 0 | 0：MIPI CSI0<br>1: MIPI CSI1 |
| -h/--height | 分辨率高 | 1080 | 无 |
| -w/--width | 分辨率宽 | 1920 | 无 |
| -d/--device_name | 设备节点 | rkispp_scale0 | 无 |
| -o/--output | 输出文件路径，未设置则不输出 | 无 | 无 |
| -c/--frame_cnt | 输出帧数，设置-1 不限制。<br>未设置输出路径不生效。 | -1 | 无 |
| -?/--help | 显示帮助信息 | 无 | 无 |

使用 rkmedia_vi_get_frame_test 命令测试如下所示：

```
rkmedia_vi_get_frame_test -a /etc/iqfiles/ -o 1080.nv12
```

“-a /etc/iqfiles/”正点原子的两款 MIPI 摄像头的 aiq 配置文件放到/etc/iqfiles/目录里，所以我们要指定从这个目录去获取摄像头的配置文件，“-o 1080.nv12”指定的输出文件为 1080.nv12。如果是接了双目摄像头需要加参数“I”去指定那一个摄像头录制。运行命令如下图所示：

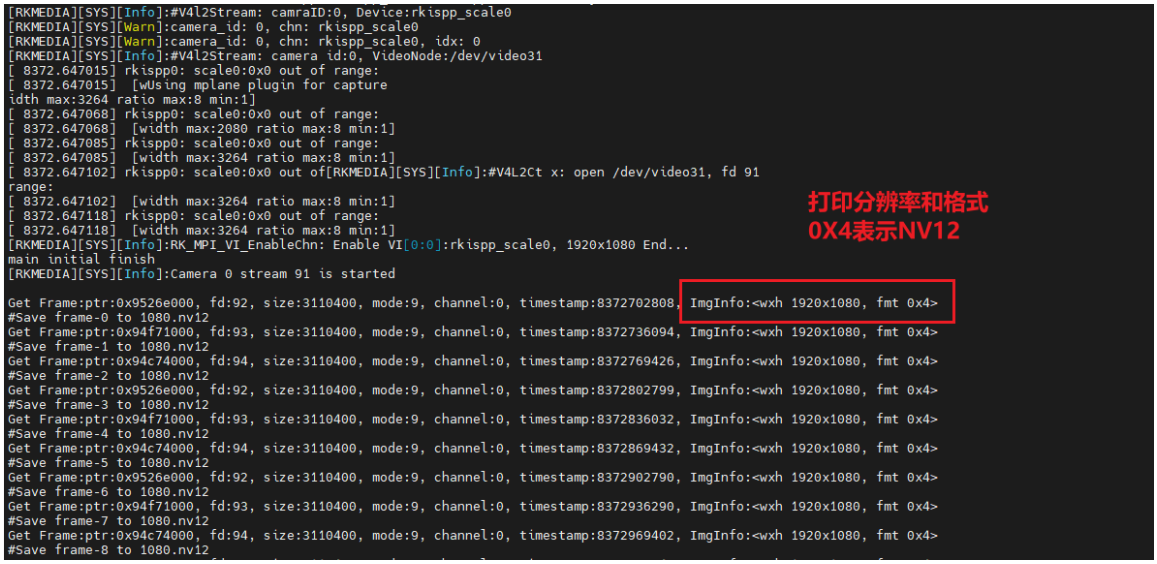



图 5.2.5.1 rkmedia_vi_get_frame_test 运行打印信息图

没有指定输出帧数需要按“Ctrl+C”结束测试程序。通过 ADB 把文件拷贝到 Ubuntu 系统里面，运行以下命令进行测试：(Ubuntu 系统里面必须要安装 ffplay 命令)

```
ffplay -f rawvideo -pix_fmt nv12    -video_size 1920x1080 1080.nv12
```

### 5.2.6 rkmedia_vi_venc_test

使用此命令前要先退出 QT 界面(点击设置→退出)，获取摄像头的数据，通过 VENC 进行编码成 H264/H265/MJPEG 格式的文件(VENC 是 rkmedia 一个编码模块)，下表是此命令的参数选项(VI→VENC)：

| 选项 | 描述 | 默认值 | 备注 |
|---|---|---|---|
| -a/--aiq | 启动 ISP 功能，如果测试的摄像头为 MIPI 摄像头基本都需要开启 ISP 功能，参数为 aiq 文件所在文件夹路径。 | /oem/etc/iqfiles/ | 无 |
| -M | ISP 下的 multictx 开关 | 0 | 0：表示关闭<br>1：开启 |
| -I | ISP 下的摄像头切换 | 0 | 0：MIPI CSI0<br>1: MIPI CSI1 |
| -h/--height | 分辨率高 | 1080 | 无 |
| -w/--width | 分辨率宽 | 1920 | 无 |
| -d/--device_name | 设备节点 | rkispp_scale0 | 无 |
| -o/--output | 输出文件路径，未设置则不输出 | 无 | 无 |
| -e/--encode | 编码格式：h264/h265/mjpeg | H264 | 无 |
| -c/--frame_cnt | 输出帧数，设置-1 不限制。<br>未设置输出路径不生效。 | -1 | 无 |
| -?/--help | 显示帮助信息 | 无 | 无 |

使用 rkmedia_vi_venc_test 命令测试如下所示：

```
rkmedia_vi_venc_test -a /etc/iqfiles/ -o output.h264
```

“-a /etc/iqfiles/”正点原子的两款 MIPI 摄像头的 aiq 配置文件放到/etc/iqfiles/目录里，所以我们要指定从这个目录去获取摄像头的配置文件，“-o output.h264”指定的输出文件为 output.h264。如果是接了双目摄像头需要加参数“I”去指定那一个摄像头录制，可以使用此命令去录制 4K 视频只需要添加参数“-h 2160 -w 3840”。运行命令如下图所示：

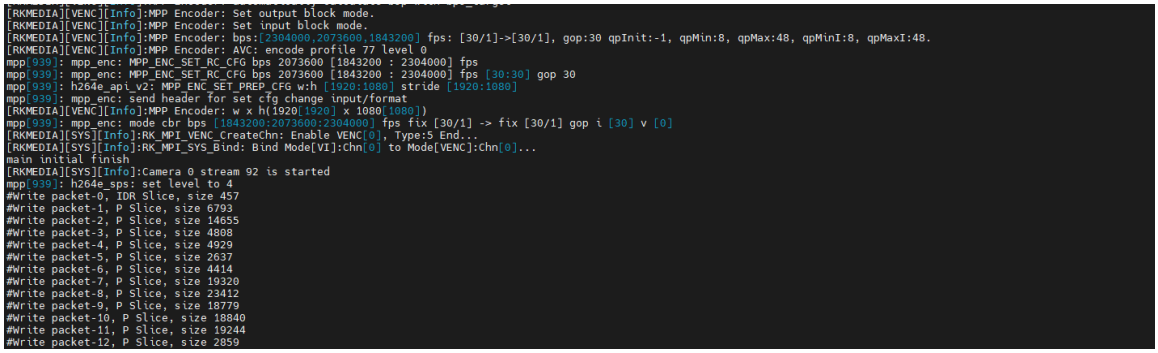



图 5.2.6.1 rkmedia_vi_venc_test 运行打印信息图

从图中可以得知视频的默认帧数为 30 帧，没有指定输出帧数需要按“Ctrl+C”结束测试程序。通过 ADB 命令把文件拷贝到 Ubuntu 系统下，使用 ffplay 进行播放：

```
ffplay output.h264
```

- rkmedia_vi_vo_test

使用此命令前要先退出 QT 界面(点击设置→退出)，获取摄像头的数据，经过 RGA 处理，显示到屏幕上。本例程需要先退出 QT 综合界面(设置→退出)，屏幕为 720 屏。下表是此命令的参数选项：

| 选项 | 描述 | 默认值 | 备注 |
|---|---|---|---|
| -a/--aiq | 启动 ISP 功能，如果测试的摄像头为 MIPI 摄像头基本都需要开启 ISP 功能，参数为 aiq 文件所在文件夹路径。 | /oem/etc/iqfiles/ | 无 |
| -M | ISP 下的 multictx 开关 | 0 | 0：表示关闭<br>1：开启 |
| -I | ISP 下的摄像头切换 | 0 | 0：MIPI CSI0<br>1: MIPI CSI1 |
| -h | 显示分辨率高 | 1280 | 无 |
| -w | 显示分辨率宽 | 720 | 无 |
| -?/--help | 显示帮助信息 | 无 | 无 |

使用 rkmedia_vi_vo_test 命令测试如下所示：

```
rkmedia_vi_vo_test -a /etc/iqfiles/
```

“-a /etc/iqfiles/”正点原子的两款 MIPI 摄像头的 aiq 配置文件放到/etc/iqfiles/目录里，所以我们要指定从这个目录去获取摄像头的配置文件。运行命令后可以看出摄像头的数据已经显示到屏幕上，默认为 30fps。按“Ctrl+C”结束测试程。

### 5.2.7 rkmedia_venc_jpeg_test

使用此命令前要先退出 QT 界面(点击设置→退出)，获取摄像头的数据，经过 VENC 编码，在叠加 RGA 模块设置位图，简单说就是把摄像头的数据给 VENC 进行编码，用 RGA 添加彩条，保存为 jpeg 格式的图片。下表是此命令的参数选项：

| 选项 | 描述 | 默认值 | 备注 |
| --- | --- | --- | --- |
| -a/--aiq | 启动 ISP 功能，如果测试的摄像头为 MIPI 摄像头基本都需要开启 ISP 功能，参数为 aiq 文件所在文件夹路径。 | /oem/etc/iqfiles/ | 无 |
| -M | ISP 下的 multictx 开关 | 0 | 0：表示关闭<br>1：开启 |
| -I | ISP 下的摄像头切换 | 0 | 0：MIPI CSI0<br>1: MIPI CSI1 |
| -h/--width | 输出分辨率宽 | 480 | 无 |
| -w/--height | 输出分辨率宽 | 720 | 无 |
| -W/--Width | 输入分辨率宽 | 1080 | 无 |
| -H/--Height | 输入分辨率高 | 1920 | 无 |
| -o | 输出文件夹的路径 | /tmp/ | 无 |
| -?/--help | 显示帮助信息 | 无 | 无 |

使用 rkmedia_venc_jpeg_test 命令测试如下所示：

```
rkmedia_venc_jpeg_test -a /etc/iqfiles/
```

“-a /etc/iqfiles/”正点原子的两款 MIPI 摄像头的 aiq 配置文件放到/etc/iqfiles/目录里，所以我们要指定从这个目录去获取摄像头的配置文件。运行命令后，就会有以下打印信息，如下图所示：

```
Had init the rga dev ctx = 0x30aa8
rga_api version 1.3.0_[11] (RGA is compiling with meson base: $PRODUCT_BASE)
[RKMEDIA][SYS][Info]:ParseMediaConfigFromMap: rect_x = 0, rect_y = 0, rect.w = 720, rect.h = 480
mpp[906]: mpp_rt: NOT found ion allocator
mpp[906]: mpp_rt: found drm allocator
mpp[906]: mpp_info: mpp version: unknown mpp version for missing VCS info
[RKMEDIA][VENC][Info]:MPP Encoder: MPPConfig: cfg init sucess!
[RKMEDIA][VENC][Info]:MPP Encoder[JPEG]: config for JPEG...
[RKMEDIA][VENC][Info]:MPP Encoder: rect_x use default value:0
[RKMEDIA][VENC][Info]:MPP Encoder: rect_y use default value:0
[RKMEDIA][VENC][Info]:MPP Encoder[JPEG]: rotaion = 0
[RKMEDIA][VENC][Info]:MPP Encoder[JPEG]: Set output block mode.
[RKMEDIA][VENC][Info]:MPP Encoder[JPEG]: Set input block mode.
mpp[906]: mpp_enc: MPP_ENC_SET_RC_CFG bps 0 [0 : 0] fps [30:30] gop 0
mpp[906]: jpege_api_v2: jpege_proc_jpeg_cfg qf_min out of range, default set 1
mpp[906]: jpege_api_v2: jpege_proc_jpeg_cfg qf_max out of range, default set 99
[RKMEDIA][VENC][Info]:MPP Encoder[JPEG]: w x h(720[720] x 480[480]), qfactor:50
[RKMEDIA][SYS][Info]:RK_MPI_VENC_CreateChn: Enable VENC[0], Type:7 End...
[RKMEDIA][SYS][Info]:RK_MPI_SYS_Bind: Bind Mode[VI]:Chn[1] to Mode[VENC]:Chn[0]...
main initial finish
#Usage: input 'quit' to exit programe!
peress any other key to capture one picture to file
[RKMEDIA][SYS][Info]:Camera 0 stream 91 is started
[ 6941.632247] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
```

图 5.2.7.1 rkmedia_venc_jpeg_test 打印信息

按“ENTER”键即可实现 jpeg 格式的截图，结束程序可以先按“Ctrl+C”然后按“ENTER”键即可结束程序。结束程序的打印如下图所示：

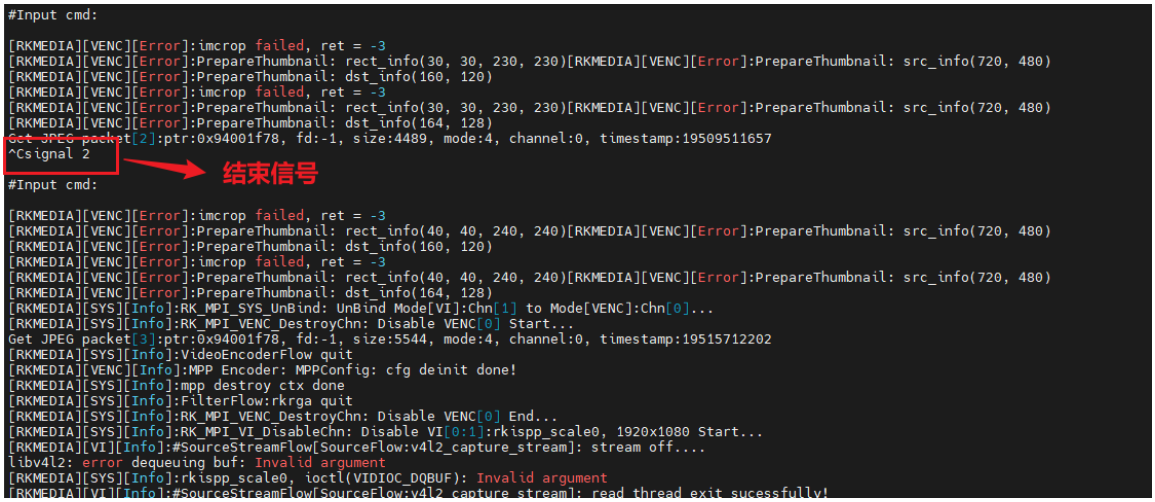



图 5.2.7.2 结束 rkmedia_venc_jpeg_test 打印信息图

结束后，在/tmp/目录下生成很多 test_jpegX.jpeg 文件，拷贝到 Ubuntu 查看即可。

### 5.2.8 rkmedia_vi_venc_rtsp_test

使用此命令前需要几个条件：

- 退出 QT 界面
- 安装 vlc 软件，Windows 系统下安装 vlc 软件，安装包位于开发板光盘 A→ 04、软件→ vlc-3.0.18-win64.exe。Ubuntu 下运行此命令：sudo apt install vlc。
- 开发板要接上网线，电脑的网络和开发板的网络处于同一个网段。

使用 VI 模块去获取摄像头的数据，经过 VENC 编码，在使用 rtsp 进行数据的推流。下表是此命令的参数选项：

| 选项 | 描述 | 默认值 | 备注 |
|---|---|---|---|
| -a/--aiq | 启动 ISP 功能，如果测试的摄像头为 MIPI 摄像头基本都需要开启 ISP 功能，参数为 aiq 文件所在文件夹路径。 | /oem/etc/iqfiles/ | 无 |
| -M | ISP 下的 multictx 开关 | 0 | 0：表示关闭<br>1：开启 |
| -I | ISP 下的摄像头切换 | 0 | 0：MIPI CSI0<br>1: MIPI CSI1 |
| -h/--width | 输出分辨率高 | 1080 | 无 |
| -w/--height | 输出分辨率宽 | 1920 | 无 |
| -d | 设备节点 | rkispp_scale0 | rkispp_scale0<br>rkispp_scale1<br>rkispp_scale2 |
| -e | 编码格式 | h264 | h264/h265 |
| -?/--help | 显示帮助信息 | 无 | 无 |

需要知道当前开发板的 IP 地址，使用 ifconfig 命令去获取 IP 地址，比如笔者这边的 IP 地址为 192.168.6.111。使用 rkmedia_vi_venc_rtsp_test 命令测试如下所示：

```
rkmedia_vi_venc_rtsp_test -a /etc/iqfiles/
```

运行后有以下数据打印，如图所示：

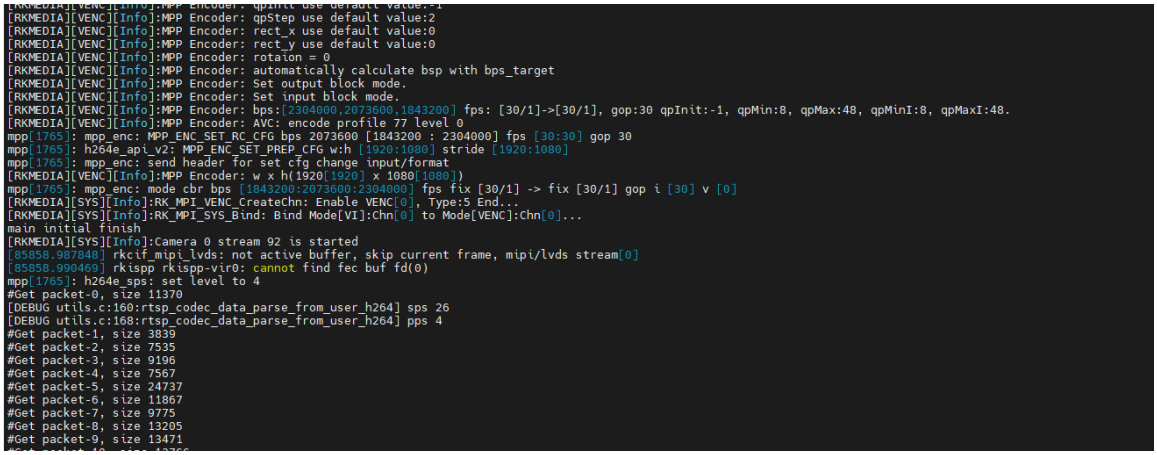



图 5.2.8.1 rkmedia_vi_venc_rtsp_test 打印信息

上图可以看出有分辨率的大小和 fps 帧数，可以使用 vlc 软件，去获取摄像头的数据，打开 vlc 软件(笔者是在 Ubuntu 下打开的，Windows 上的用法差不多)，如下图所示：

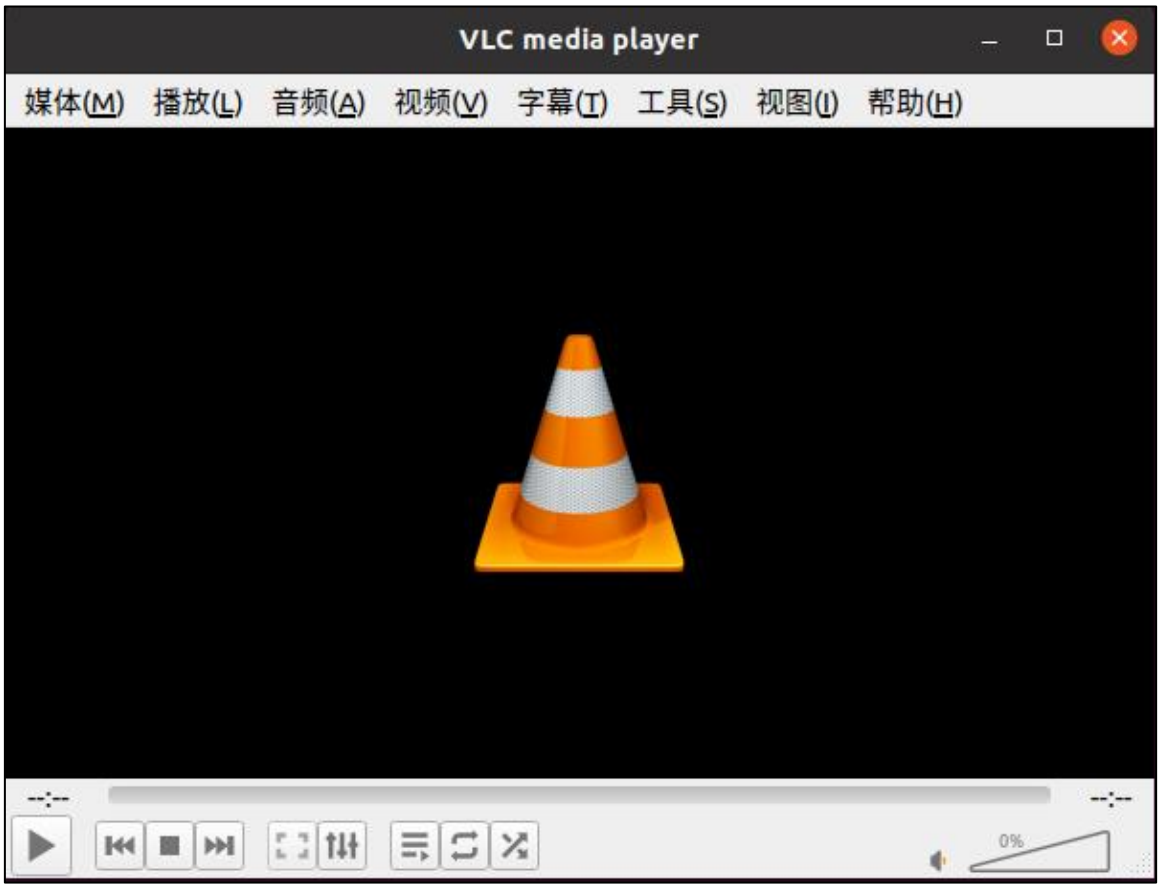



图 5.2.8.2 Ubuntu 系统下开启 vlc

图 5.2.8.2 中点击“媒体”进入“打开媒体”，如下图所示：

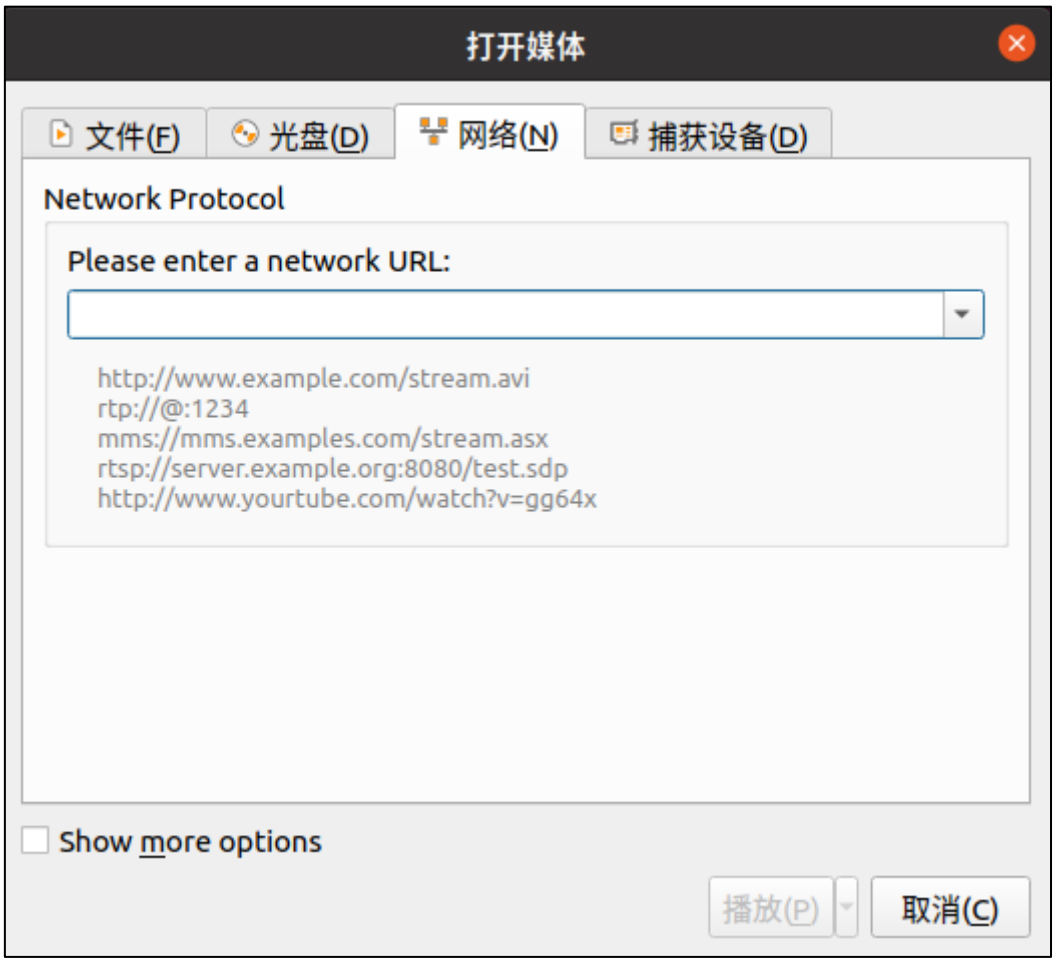



图 5.2.8.3 打开媒体

在图 4.2.8.3 中“please enter a network URL:”框架中输入以下链接：

```
rtsp://192.168.6.111/live/main_stream
```

这里的 IP 地址为开发板上的 IP 地址，如下图所示：


图 5.2.8.4 拉流设置

设置好拉流的地址后，直接点击播放，即可出现摄像头的播放数据，如下图所示：

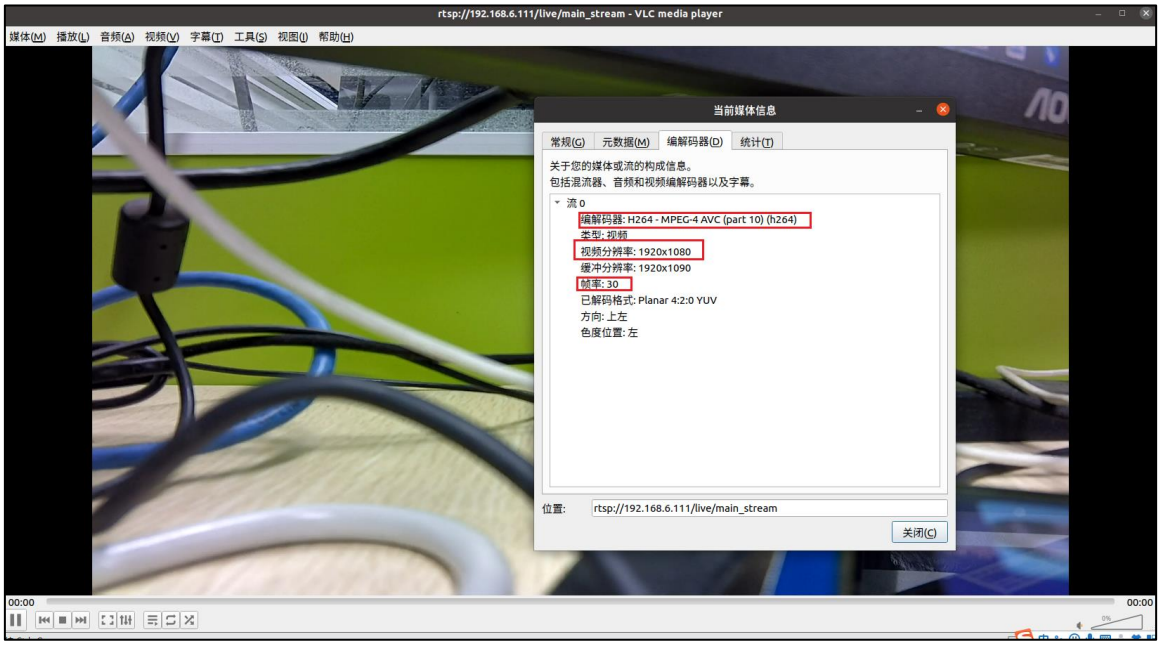



图 5.2.8.5 拉流视频播放

从上图可以看出，拉流的视频分辨率为：1920x1080@30，数据格式为 H264。大家可以修

改分辨率。

### 5.2.9 rkmedia_vi_vo_test

使用此命令前要先退出 QT 界面(点击设置→退出)，使用 VI 获取摄像头的数据，通过接口发送到 VO，就能实现显示到屏幕上。下表是此命令的参数选项(VI→VO):

| 选项 | 描述 | 默认值 | 备注 |
| --- | --- | --- | --- |
| -a/--aiq | 启动 ISP 功能，如果测试的摄像头为 MIPI 摄像头基本都需要开启 ISP 功能，参数为 aiq 文件所在文件夹路径。 | /oem/etc/iqfiles/ | 无 |
| -M | ISP 下的 multictx 开关 | 0 | 0：表示关闭<br>1：开启 |
| -I | ISP 下的摄像头切换 | 0 | 0：MIPI CSI0<br>1: MIPI CSI1 |
| -h | 分辨率高 | 1280 | 无 |
| -w | 分辨率宽 | 720 | 无 |
| -?/--help | 显示帮助信息 | 无 | 无 |

使用命令如下所示：

```
rkmedia_vi_vo_test -a /etc/iqfiles/
```

运行结果如下图所示：

```
[RKMEDIA][SYS][Info]:RKAIQ: model(rkispp1): ispp_info(1): ispp-subdev entity name: /dev/v4l-subdev1
[RKMEDIA][SYS][Info]:#V4l2Stream: camraID:0, Device:rkispp_scale0
[RKMEDIA][SYS][Warn]:camera_id: 0, chn: rkispp_scale0
[RKMEDIA][SYS][Warn]:camera_id: 0, chn: rkispp_scale0, idx: 0
[RKMEDIA][SYS][Info]:#V4l2Stream: camera id:0, VideoNode:/dev/video31
[14287.982656] rkispp0: scale0:0x0 out of range:
[14287.982656]  [wUsing mplane plugin for captiure
dth max:3264 ratio max:8 min:1]
[14287.982723] rkispp0: scale0:0x0 out of range:
[14287.982723]  [width max:2080 ratio max:8 min:1]
[14287.982741] rkispp0: scale0:0x0 out of range:
[14287.982741]  [width max:3264 ratio max:8 min:1]
[14[RKMEDIA][SYS][Info]:#V4L2Ctx2: open /dev/video31, fd 91
87.982757] rkispp0: scale0:0x0 out of r[RKMEDIA][SYS][Info]:Opened DaRM device /dev/dri/card0: drivner rockchip version 2.0.0.
ge:
[14287.982757]  [width max:3264 ratio max:8 min:1]
[14287.982774] rkispp0: scale0:0x0 out of range:
[14287.982774]  [width max:3264 ratio max:8 min:1]
[RKMEDIA][SYS][Info]:RK_MPI_VI_EnableChn: Enable VI[0:0]:rkispp_scale0, 1920x1080 End...
[RKMEDIA][SYS][Info]:RK_MPI_RGA_CreateChn: Enable RGA[0], Rect<0,0,1920,1080> Start...
[RKMEDIA][SYS][Info]:FilterFlow:rkrga: Enable BufferPool! memtype:hw_mem, memcnt:3
[RKMEDIA][SYS][Info]:RK_MPI_RGA_CreateChn: Enable RGA[0], Rect<0,0,1920,1080> End...
[RKMEDIA][SYS][Info]:RK_MPI_VO_CreateChn: Enable VO[0] Start...
[RKMEDIA][SYS][Info]:conn id : 56, enc id: 55, crtc id: 53, plane id: 52, w/h: 1080,1920, fps: 32
[RKMEDIA][SYS][Info]:RK_MPI_VO_CreateChn: Enable VO[0] End!
#Bind VI[0] to RGA[0]....
[RKMEDIA][SYS][Info]:RK_MPI_SYS_Bind: Bind Mode[VI]:Chn[0] to Mode[RGA]:Chn[0]...
# Bind RGA[0] to VO[0]....
[RKMEDIA][SYS][Info]:RK_MPI_SYS_Bind: Bind Mode[RGA]:Chn[0] to Mode[VO]:Chn[0]...
main initial finish
[RKMEDIA][SYS][Info]:Camera 0 stream 91 is started
[14288.107488] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
[RKMEDIA][SYS][Error]:FilterFlow:rkrga: buffer_pool get null buffer!
```

图 5.2.9.1 运行 rkmedia_vi_vo_test 结果

接着我们的显示屏，就能显示摄像头的实时数据。

### 5.2.10 rkmedia_vi_double_cameras_switch_test

使用此命令前要先退出 QT 界面(点击设置→退出)，使用 2 个 VI 获取摄像头的数据(双目摄像头的测试需要接两个摄像头)，通过接口发送到 VMIX，VMIX 就会把合成一帧数据，然后把数据发送到 VO 里，这样就能够显示两个摄像头的数据。下表是此命令的参数选项(VI→VMIX→VO)：

| 选项 | 描述 | 默认值 | 备注 |
| --- | --- | --- | --- |

| | | | |
|---|---|---|---|
| -a/--aiq | 启动 ISP 功能，如果测试的摄像头为 MIPI 摄像头基本都需要开启 ISP 功能，参数为 aiq 文件所在文件夹路径。 | /oem/etc/iqfiles/ | 无 |
| -h | 输出视频分辨率高 | 1280 | 无 |
| -w | 输出视频分辨率宽 | 720 | 无 |
| -W | 输入视频分辨率宽 | 1920 | 无 |
| -H | 输入视频分辨率高 | 1080 | 无 |
| -u | UI z 层高度，取值范围[0,1] | 1 | 无 |
| -?/--help | 显示帮助信息 | 无 | 无 |

使用命令如下所示：

```
rkmedia_vi_double_cameras_test -a /etc/iqfiles/ -u 0
```

注意：需要选择图层 0，这个和 drm 框架相关。运行结果如下打印：

```
[RKMEDIA][SYS][Info]:Camera 0 stream 175 is started
  284.593850] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  284.607545] rockchip-mipi-dphy-rx ff4b8000.csi-dphy: stream on:1
  284.607610] rockchip-mipi-dphy-rx: data_rate_mbps 1188
[RKMEDIA][SYS][Info]:Camera 1 stream 180 is started
[RKMEDIA][SYS][Error]:FilterFlow:rkrga: buffer_pool get null buffer!
[RKMEDIA][SYS][Error]:FilterFlow:rkrga: buffer_pool get null buffer!
  284.893888] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  284.943182] rkisp1: tx stream:4 lose frame:4, isp state:0x204 frame:1
  284.983273] rkisp1: tx stream:4 lose frame:5, isp state:0x204 frame:2
  284.993824] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.027323] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.104339] rkisp1: tx stream:4 lose frame:8, isp state:0x201 frame:3
  285.127241] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.183187] rkisp1: tx stream:4 lose frame:10, isp state:0x201 frame:4
  285.193978] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.263215] rkisp1: tx stream:4 lose frame:12, isp state:0x201 frame:7
[RKMEDIA][SYS][Error]:FilterFlow:rkrga: buffer_pool get null buffer!
  285.327306] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.343172] rkisp1: tx stream:4 lose frame:14, isp state:0x201 frame:8
[RKMEDIA][SYS][Error]:FilterFlow:rkrga: buffer_pool get null buffer!
  285.393861] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.423208] rkisp1: tx stream:4 lose frame:16, isp state:0x201 frame:10
  285.460739] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.503240] rkisp1: tx stream:4 lose frame:18, isp state:0x201 frame:12
  285.527271] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.583297] rkisp1: tx stream:4 lose frame:20, isp state:0x201 frame:14
  285.660552] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.663244] rkisp1: tx stream:4 lose frame:22, isp state:0x201 frame:16
  285.727272] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.784321] rkisp1: tx stream:4 lose frame:25, isp state:0x201 frame:20
[RKMEDIA][SYS][Error]:FilterFlow:rkrga: buffer_pool get null buffer!
  285.860844] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  285.863277] rkisp1: tx stream:4 lose frame:27, isp state:0x201 frame:22
[RKMEDIA][SYS][Error]:FilterFlow:rkrga: buffer_pool get null buffer!
  285.983333] rkisp1: tx stream:4 lose frame:30, isp state:0x201 frame:25
  285.993824] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  286.060730] rkcif_mipi_lvds: not active buffer, skip current frame, mipi/lvds stream[0]
  286.063243] rkisp1: tx stream:4 lose frame:32, isp state:0x201 frame:27
```

图 5.2.10 rkmedia_vi_double_cameras_test 运行打印

# 第六章 buildroot 使用

RV1126 的文件系统是使用 buildroot 构建的，本章节基于 RV1126 开发板教大家如何修改或者编译 buildroot 软件包和添加自己的软件进入 buildroot 里。具体的使用方法请看【正点原子】Buildroot 用户手册中文版(正点原子翻译)_V1.0.pdf 文件。

## 6.1 buildroot 的使用技巧

学习过 buildroot 的需要先使能配置文件，就是在 buildroot 目录下运行“make xxxx_defconfig”,然后就会在本目录下生成“.config”文件，此文件就是 buildroot 的配置文件。在 RV1126 上是使用“envsetup.sh”脚本进生成“.config”文件(使用方法请看 4.2 小节)。运行成功后，此时的终端相对于运行“make xxxx_defconfig”，我们就可以直接使用 make 命令进行操作 buildroot(使用 buildroot 前，必须使用配置文件)。

- 列出 package 包

Buildroot 的编译目标就是 package 的数量包，所以我们需要知道有多个 package 包的数量，运行命令如下所示：

```
make show-targets
```

运行结果如下图所示：


图 6.1.1 列出 package 包数量

图 6.1.1 中就是 RV1126 的文件系统需要软件包，数量越多说明文件系统越大，我们就可以添加或者删减软件包来精简文件系统。

- 删除或者添加 package 包

使用此命令进行配置 package 包，命令如下所示：

```
make menuconfig
```

运行结果如下图所示：

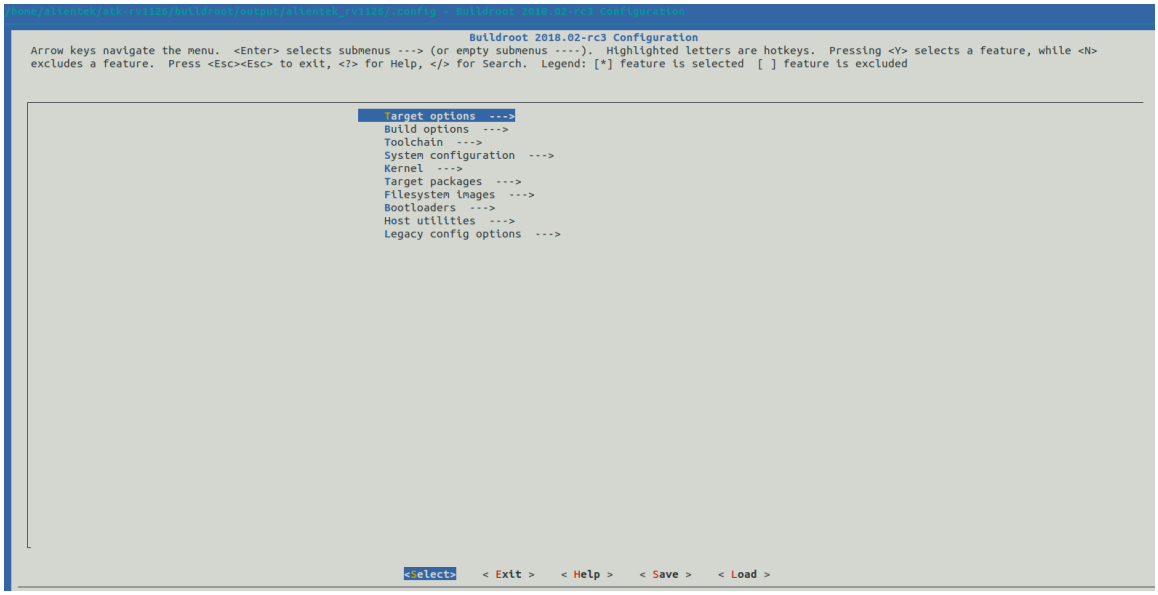



图 6.1.2 buildroot 的图形化界面

进入图形化界面后，我们可以在这里添加或者删除 package 包，比如：删除 rkwifibt，配置路径如下：

```
Target packages  →
    [*] Rockchip BSP packages  →
        [ ]  rkwifibt  →
```

配置如下图所示：

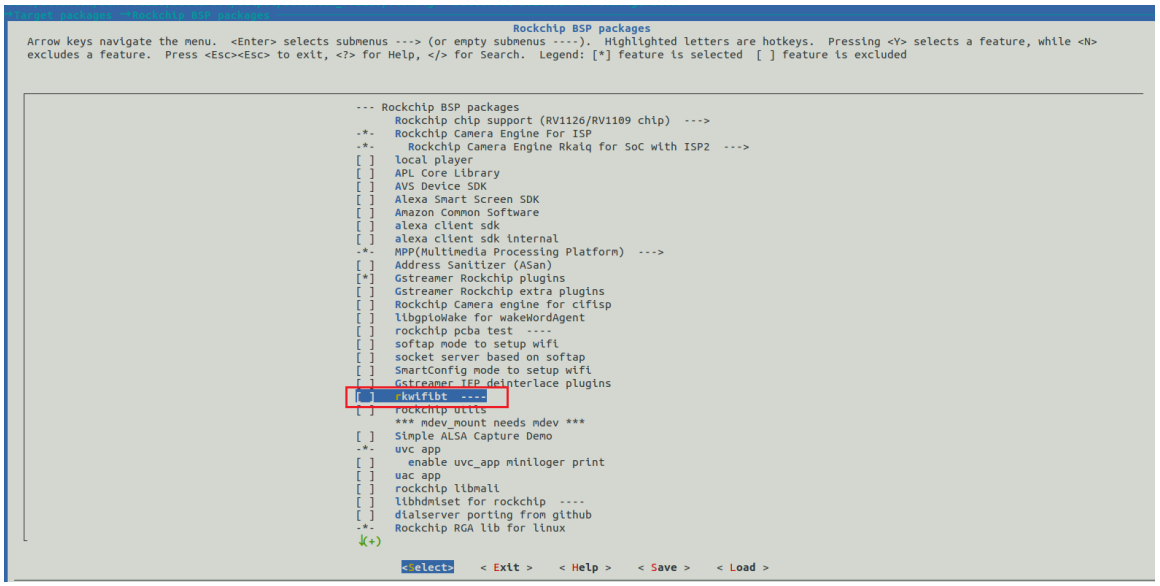



图 6.1.3 删除 rkwifibt 包

配置完成后，可以使用“make show-targets”命令查看是否取消 rkwifibt 成功。如果真的需要取消 rkwifibt，运行此命令“make savedefconfig”保存“.config”文件到“buildroot/configs/alientek_rv1126_defconfig”文件里，下次使能配置文件就没有 rkwifibt。

有时候我们会出现编译 buildroot 的编译出错，是因为内存不足，可以先单独编译 package 包或者增大内存。内存不够的可以修改 Ubuntu 系统下的 swap 空间。

## 6.2 package 包的命令使用

### 6.2.1 package 包命令列表

Buildroot 提供了可以单独下载配置、补丁和编译 package 包。运行以下命令列出 package 的使用命令。命令如下所示：

```
make help
```

运行结果如下图所示：

```
Package-specific:
  <pkg>                  - Build and install <pkg> and all its dependencies
  <pkg>-source           - Only download the source files for <pkg>
  <pkg>-extract          - Extract <pkg> sources
  <pkg>-patch            - Apply patches to <pkg>
  <pkg>-depends          - Build <pkg>'s dependencies
  <pkg>-configure        - Build <pkg> up to the configure step
  <pkg>-build            - Build <pkg> up to the build step
  <pkg>-show-depends     - List packages on which <pkg> depends
  <pkg>-show-rdepends    - List packages which have <pkg> as a dependency
  <pkg>-graph-depends    - Generate a graph of <pkg>'s dependencies
  <pkg>-graph-rdepends   - Generate a graph of <pkg>'s reverse dependencies
  <pkg>-dirclean         - Remove <pkg> build directory
  <pkg>-reconfigure      - Restart the build from the configure step
  <pkg>-rebuild          - Restart the build from the build step
```

图 6.1.4 package 命令使用

从上图 6.1.4 就是“package”包使用命令。下表如何使用此命令：

| 命令/目标 | 说明 | 例子 |
|---|---|---|
| <pkg> | 编译和构建<pkg>包和<pkg>的依赖项 | make rkwifibt |
| <pkg>-source | 下载源码到目录里 | make rkwifibt-source |
| <pkg>-extract | 将源码提取到软件包构建目录 | make rkwifibt-extract |
| <pkg>-patch | 打补丁到软件包构建目录下 | make rkwifibt-patch |
| <pkg>-depends | 编译<pkg>包的依赖项 | make rkwifibt-depends |
| <pkg>-configure | 配置编译前的命令(下载、提取、补丁和编译依赖项) | make rkwifibt-configure |
| <pkg>-build | 运行编译命令 | make rkwifibt-build |
| <pkg>-show-depends | 列出<pkg>包的依赖项 | make rkwifibt-show-depends |
| <pkg>-show-rdepends | 列出<pkg>包作为依赖项的软件包 | make rkwifibt-show-rdepends |
| <pkg>-graph-depends | 以 PDF 的形式生成<pkg>包的依赖项 | make rkwifibt-graph-depends |
| <pkg>-graph-rdepends | 以 PDF 的形式生成<pkg>包为依赖项的软件包 | make rkwifibt-graph-rdepends |
| <pkg>-dirclean | 删除整个软件包构建目录 | make rkwifibt-dirclean |
| <pkg>-reconfigure | 重新运行配置命令 | make rkwifibt-reconfigure |
| <pkg>-rebuild | 重新运行编译命令 | make rkwifibt-rebuild |

### 6.2.2 package 下载测试

我们首先需要使用“make show-targets”命令列出 package 包的名字，比如笔者这里测试需要下载 iperf3 命令，需要在源码目录下，运行以下命令删除 iperf3 源码包(下载好了不会再次下载)，命令如下所示：

```
rm buildroot/dl/iperf-3.6.tar.gz
```

接着可以使用命令，下载 iperf3 软件。命令如下所示：

```
make iperf3-source
```

运行结果如下图所示：

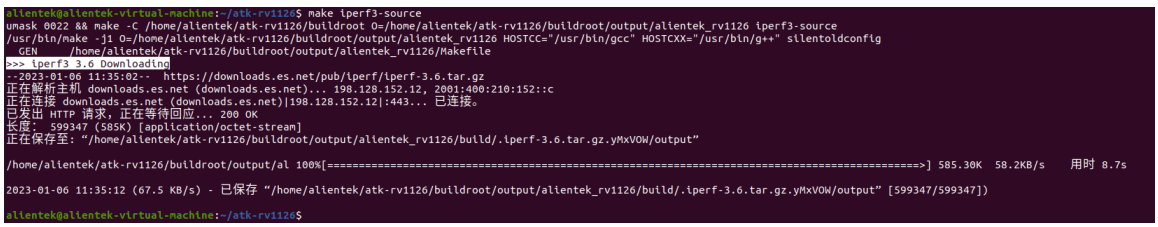



图 6.2.2.1 下载 iperf3 的源码

注意图中的下载路径，如果下载路径是从“github”网站下载，有可能下载不了，国内的网络不能访问。下载完成后保存到“buildroot/dl/”目录下。

### 6.2.3 package 提取

“提取”其实就是解压源码包到编译目录下，命令如下所示：

```
make iperf3-extract
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ make iperf3-extract
umask 0022 && make -C /home/alientek/atk-rv1126/buildroot O=/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126 iperf3-extract
>>> iperf3 3.6 Extracting
gzip -d -c /home/alientek/atk-rv1126/buildroot/dl/iperf-3.6.tar.gz | /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/host/bin/tar --strip-components=1 -C /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/build/iperf3-3.6   -xf -
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 6.2.3.1 提取 iperf3 源码包

提取成功后，在“buildroot/output/alientek_rv1126/build”目录下生成“iperf-3.6”目录，可以跳转到“iperf-3.6”目录查看是否解压成功。结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126/buildroot/output/alientek_rv1126/build/iperf3-3.6$ ls
aclocal.m4    config     configure.ac  docs       INSTALL         LICENSE      Makefile.in   README.md      src
bootstrap.sh  configure  contrib       examples   iperf3.spec.in  Makefile.am  make_release  RELEASE_NOTES  test_commands.sh
alientek@alientek-virtual-machine:~/atk-rv1126/buildroot/output/alientek_rv1126/build/iperf3-3.6$
```

图 6.2.3.2 查看 iperf3 源码

### 6.2.4 package 打补丁

给 package 包打补丁的命令代码如下所示：

```
make ffmpeg-patch
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ make iperf3-parch
umask 0022 && make -C /home/alientek/atk-rv1126/buildroot O=/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126 iperf3-parch
make[1]: *** 没有规则可制作目标“iperf3-parch”。 停止。
make: *** [/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/Makefile:16：_all] 错误 2
alientek@alientek-virtual-machine:~/atk-rv1126$
alientek@alientek-virtual-machine:~/atk-rv1126$
alientek@alientek-virtual-machine:~/atk-rv1126$ make ffmpeg-patch
umask 0022 && make -C /home/alientek/atk-rv1126/buildroot O=/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126 ffmpeg-patch
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 6.2.4 ffmpeg 打补丁

如果没有补丁的就会报错，比如上图中的“iperf3”打补丁，就报错了。

### 6.2.5 depends 编译软件包的依赖项

有些 package 包需要一些依赖包，比如正点原子自定义的 Qdesktop 软件包，需要依赖 QT 源码包的编译完成，后才能编译 Qdesktop 软件包，所以我们需要先把一些依赖项编译。命令如下所示：

```
make iperf3-depends
```

运行完成后如下图所示：

```
install ./include/openssl/x509err.h -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/include/openssl/x509err.h
install ./include/openssl/x509v3.h -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/include/openssl/x509v3.h
install ./include/openssl/x509v3err.h -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/include/openssl/x509v3err.h
install libcrypto.a -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/libcrypto.a
install libssl.a -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/libssl.a
link /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/libcrypto.so -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126
.1.1
link /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/libssl.so -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/ta
install libcrypto.pc -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/pkgconfig/libcrypto.pc
install libssl.pc -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/pkgconfig/libssl.pc
install openssl.pc -> /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/pkgconfig/openssl.pc
rm -rf /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/ssl
rm -f /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/bin/c_rehash
rm -f -f /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/etc/ssl/misc/{CA.pl,tsget}
rm -f -f /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/bin/openssl
rm -f -f /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/etc/ssl/misc/{CA.*,c_*}
rm -rf /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/target/usr/lib/engines-1.1
>>> openssl  Extracting
>>> openssl  Patching
>>> openssl  Configuring
>>> openssl  Building
>>> openssl  Installing to target
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 6.2.5.1 iperf3 依赖项编译

### 6.2.6 configure 配置

“配置”这里的配置包含了下载、提取、打补丁和编译依赖项，运行命令如下所示：

```
make iperf3-configure
```

运行结果如下图所示：


图 6.2.6 configure 配置

### 6.2.7 build 编译

单独编译 package 包，命令如下所示：

```
make iperf3-build
```

运行结果如下图所示：


图 6.2.7.1 编译 package 包

### 6.2.8 列出依赖项

可以单独列出 package 包，所需要的依赖项，命令如下所示：

```
make iperf3-show-depends
```

运行结果如下图所示：


图 6.2.8.1 iperf3 依赖项

在 6.2.5 小节里面，编译依赖项有 openssl 这个软件包，所以这里列出来有这个 openssl 软件包。

### 6.2.9 dirclean 清除软件包

可以单独清除 package 包，运行命令如下图所示：

```
make iperf3-dirclean
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ make iperf3-dirclean
umask 0022 && make -C /home/alientek/atk-rv1126/buildroot O=/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126 iperf3-dirclean
grep: /home/alientek/atk-rv1126/buildroot/../kernel/.config: 没有那个文件或目录
rm -Rf /home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/build/iperf3-3.6
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 6.2.9.1 清除软件包

### 6.2.10 修改软件版本

本小节教大家如何修改 buildroot 的软件包里的软件版本，比如笔者需要修改 iperf3 的版本，运行以下命令跳转到 iperf3 的配置目录：

```
cd buildroot/package/iperf3/
ls
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126/buildroot/package/iperf3$ ls
0001-disable-profiling.patch  0002-Fix-build-using-musl-libc.patch  Config.in  iperf3.hash  iperf3.mk
alientek@alientek-virtual-machine:~/atk-rv1126/buildroot/package/iperf3$
```

图 6.2.10.1 查看 iperf3

上图中有 3 个重要的文件分别为：Config.in、iperf3.hash 和 iperf3.mk。Config.in 相当于内核的 Kconfig，iperf3.hash 软件包的校验码，iperf3.mk 相对于内核的 Makefile。我们打开 iperf3.mk 文件找到下载文件的下载地址(有时候下载地址在 Config.in 里，看编写文件的作者)，打开如下图所示：

```
################################################################################
#
# iperf3
#
################################################################################

IPERF3_VERSION = 3.6
IPERF3_SITE = https://downloads.es.net/pub/iperf
IPERF3_SOURCE = iperf-$(IPERF3_VERSION).tar.gz
IPERF3_LICENSE = BSD-3-Clause, BSD-2-Clause, MIT
IPERF3_LICENSE_FILES = LICENSE

IPERF3_CONF_ENV += CFLAGS="$(TARGET_CFLAGS) -D_GNU_SOURCE"

ifeq ($(BR2_PACKAGE_OPENSSL),y)
# We intentionally don't pass --with-openssl, otherwise pkg-config is
# not used, and indirect libraries are not picked up when static
# linking.
IPERF3_DEPENDENCIES += host-pkgconf openssl
else
IPERF3_CONF_OPTS += --without-openssl
endif

$(eval $(autotools-package))
```

图 6.2.10.2 iperf3.mk 文件查看

上图可以看出来，有下载地址和 iperf3 的版本，我们可以先打开此下载链接地址“https://downloads.es.net/pub/iperf/”找到我们需要更新的版本。如下图所示：

```
iperf-3.4.tar.gz.sha256                  12-Feb-2018 22:35                83
iperf-3.4.txt.asc                        14-Feb-2018 21:19              2913
iperf-3.5.tar.gz                         28-Feb-2018 21:35            593449
iperf-3.5.tar.gz.sha256                  28-Feb-2018 21:35                83
iperf-3.6.tar.gz                         22-Jun-2018 21:28            599347
iperf-3.6.tar.gz.sha256                  22-Jun-2018 21:28                83
iperf-3.7.tar.gz                         21-Jun-2019 02:47            605708
iperf-3.7.tar.gz.sha256                  21-Jun-2019 02:47                83
iperf-3.7.txt.asc                        21-Jun-2019 19:58              2918
iperf-3.8.1.tar.gz                       10-Jun-2020 15:24            618098
iperf-3.8.1.tar.gz.sha256                10-Jun-2020 15:24                85
iperf-3.8.tar.gz                         08-Jun-2020 21:03            618032
iperf-3.8.tar.gz.sha256                  08-Jun-2020 21:03                83
iperf-3.9.tar.gz                         17-Aug-2020 18:29            622459
iperf-3.9.tar.gz.sha256                  17-Aug-2020 18:29                83
```

图 6.2.10.3 iperf3 版本查看

图 6.2.10.3 中最新支持 3.9 版本，我们就修改 iperf3.mk 里的 3.6 改为 3.9，修改结果如下图所示：

```
################################################################################
#
# iperf3
#
################################################################################

IPERF3_VERSION = 3.9
IPERF3_SITE = https://downloads.es.net/pub/iperf
IPERF3_SOURCE = iperf-$(IPERF3_VERSION).tar.gz
IPERF3_LICENSE = BSD-3-Clause, BSD-2-Clause, MIT
IPERF3_LICENSE_FILES = LICENSE

IPERF3_CONF_ENV += CFLAGS="$(TARGET_CFLAGS) -D_GNU_SOURCE"

ifeq ($(BR2_PACKAGE_OPENSSL),y)
# We intentionally don't pass --with-openssl, otherwise pkg-config is
# not used, and indirect libraries are not picked up when static
# linking.
IPERF3_DEPENDENCIES += host-pkgconf openssl
else
IPERF3_CONF_OPTS += --without-openssl
endif

$(eval $(autotools-package))
```

图 6.2.10.4 修改 iperf3 版本

修改成功后，我们就可以使用 6.2.2 小节进行下载测试，命令如下所示：

```
make iperf3-source
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ make iperf3-source
umask 0022 && make -C /home/alientek/atk-rv1126/buildroot O=/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126 iperf3-source
grep: /home/alientek/atk-rv1126/buildroot/../kernel/.config: 没有那个文件或目录
>>> iperf3 3.9 Downloading
--2023-01-06 17:20:48--  https://downloads.es.net/pub/iperf/iperf-3.9.tar.gz
正在解析主机 downloads.es.net (downloads.es.net)... 198.128.152.12, 2001:400:210:152::c
正在连接 downloads.es.net (downloads.es.net)|198.128.152.12|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度： 622459 (608K) [application/octet-stream]
正在保存至: “/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/build/.iperf-3.9.tar.gz.NMHKe8/output”

/home/alientek/atk-rv1126/buildroot/output/al 100%[=========================================================================================>] 607.87K   880KB/s    用时 0.7s

2023-01-06 17:20:50 (880 KB/s) - 已保存 “/home/alientek/atk-rv1126/buildroot/output/alientek_rv1126/build/.iperf-3.9.tar.gz.NMHKe8/output” [622459/622459])

alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 6.2.10.5 下载 3.9 版本的 iperf3

上图已经下载成功了，下载到“buildroot/dl/”目录下，还需要运行以下命令生成源码的 hash 值，命令如下所示：

```
sha256sum buildroot/dl/iperf-3.9.tar.gz
```

运行结果如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ sha256sum buildroot/dl/iperf-3.9.tar.gz
24b63a26382325f759f11d421779a937b63ca1bc17c44587d2fcfedab60ac038  buildroot/dl/iperf-3.9.tar.gz
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 6.2.10.6 生成 hash 值

跳转到“buildroot/package/iperf3/”下，去修改 iperf3.hash 文件，打开此文件如下图所示：

```
# From https://downloads.es.net/pub/iperf/iperf-3.6.tar.gz.sha256
sha256 de5d51e46dc460cc590fb4d44f95e7cad54b74fea1eba7d6ebd6f8887d75946e  iperf-3.6.tar.gz
# Locally computed
sha256 52c42914d7d79fe5e95d0d1b821556d9f06bf756ac910fe085a46d238a33e594  LICENSE
~
```

图 6.2.10.7 iperf3.hash 内容

把图中 6.2.10.7 的值替换成图 6.2.10.6 值，替换结果如下图所示：

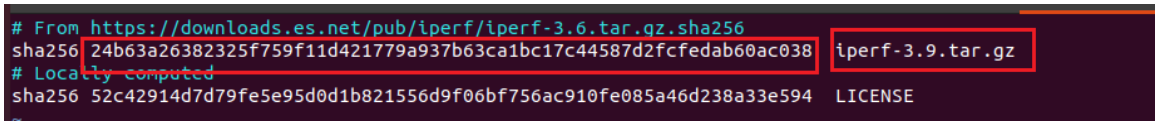



图 6.2.10.8 替换结果

上图红色框的 hash 值和图 6.2.10.6 显示的值是一样的，还需要修改软件的版本号。修改完成后就可以编译源码到文件系统里面。因为我们的 3.9 版本是没有补丁的所以需要把“buildroot/package/iperf3/”下的两个补丁删除掉。

## 6.3 添加自己的程序

本章节教大家如何添加自己的程序到 buildroot 里。

### 6.3.1 添加 test.c 文件

在源码目录下，创建一个自己的 APP 文件夹，如下命令：

```
mkdir app/mytest
```

运行结果如下图所示：

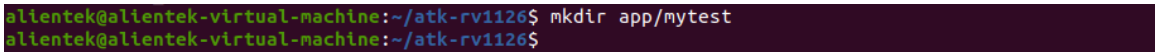



图 6.3.1.1 创建 mytest

在 mytest 目录下可以创建自己的 APP 程序，这里笔者就创建一个 test.c 的测试代码，示例代码如下所示：

示例代码 6.3.1.1 test.c 代码

```
include <stdio.h>

int main()
{
    printf("buildroot helloworld\n");
    return 0;
}
```

还是在 mytest 目录下，创建一个编译 test.c 程序的 Makefile，示例代码如下所示：

示例代码 6.3.1.2 Makefile 代码

```
Wall -Wno-deprecated
CFLAGS = $(OPT) $(OTHER)
INCDIR = -I
LIBDIR = -L
LIBS =
APP=mytest
SRCS=test.c

all:
	$(CC) -o $(APP) $(SRCS) $(CFLAGS) $(LIBDIR) $(INCDIR) $(LIBS)
clean:
```

```
        rm $(APP)
```

创建结果完成如下图所示：

```
alientek@alientek-virtual-machine:~/atk-rv1126$ ls app/mytest/
Makefile  test.c
alientek@alientek-virtual-machine:~/atk-rv1126$
```

图 6.3.1.2 mytest 的目录查看

### 6.3.2 加到 menuconfig 里

大家在配置 buildroot 的 package 包的时候，都需要在图形化配置界面上使能，所以我们的 APP 也是需要生成一个选项。在 SDK 包的源码下，打开“buildroot/package/Config.in”文件。跳转到最后一个“endmenu”下添加如下示例代码(注意：必须在此文件的最后一个 endmenu 前添加)。代码如下所示：

```
menu "mytest"
    source "package/mytest/Config.in"
endmenu
```

添加结果如下图所示：

```
2060 menu "mytest"
2061         source "package/mytest/Config.in"
2062 endmenu
2063
2064 endmenu
```

图 6.3.2.1 添加结果

第 2061 行，里面使用了 source 引用了“package/mytest/Config.in”文件，所以我们需要创建此文件，命令如下所示：

```
mkdir buildroot/package/mytest/
```

创建目录后(mytest 就是在 buildroot 下 package 包的名字)，我们需要再 mytest 目录创建“Config.in”文件，内容如下所示：

```
config BR2_PACKAGE_MYTEST
bool "mytest"
help
  This is a demo to add MYtest.
```

添加结果如下图所示：

```
config BR2_PACKAGE_MYTEST
bool "mytest"
help
  This is a demo to add MYtest.
~
```

图 6.3.2.2 Config.in 文件

### 6.3.3 APP 的版本和编译规则

在上个小节里面我们已经把 APP 添加到配置选项中了，此时的 buildroot 还是不知道如何下载、提取、编译等等，我们需要添加配置文件告诉 buildroot 如何操作，需要创建“buildroot/package/mytest/mytest.mk”，名字必须为 mytest.mk。把以下的示例代码拷贝到 mytest.mk 里。

示例代码 6.3.3.1 mytest.mk 代码

```
1  MYTEST_VERSION:= 1.0.0
2  MYTEST_SITE:= $(TOPDIR)/../app/mytest
3  MYTEST_SITE_METHOD:=local
4  MYTEST_INSTALL_TARGET:=YES
5
6  define MYTEST_BUILD_CMDS
7      $(MAKE) CC="$(TARGET_CC)" LD="$(TARGET_LD)" -C $(@D) all
8  endef
9
10 define MYTEST_INSTALL_TARGET_CMDS
11     $(INSTALL) -D -m 0755 $(@D)/mytest $(TARGET_DIR)/bin
12 endef
13
14 define MYTEST_PERMISSIONS
15     /bin/mytest f 4755 0 0 - - - - -
16 endef
17
18 $(eval $(generic-package))
```

此文件就是告诉 buildroot 是如何编译、下载、拷贝等等。大写的宏都是以“MYTEST”开头的，是根据我们 package 包的名字决定的。

第 1 行，_VERSION 结尾的变量是源码的版本。

第 2 行，_SITE 结尾的变量是源码下载的地址。

第 3 行，_SITE_METHOD 结尾的变量是下载源码方法，local 为本地下载。

第 4 行和第 10~12 行，自动执行安装，把我们的 mytest 可以运行的文件拷贝 bin 目录下。

第 6 行，_BUILD_CMDS 结尾的变量会在 buildroot 框架编译的时候执行，用于给源码的 Makefile 传递编译选项和链接选项，调用源码的 Makefile。

第 14~16 行，给文件的权限。

第 18 行，这个函数会把整个.mk 文件构建成脚本。

### 6.3.4 测试 APP

前面几个小节里面已经配置好了我们的 mytest 了，接着需要在 menuconfig 里面开启 mytest。如下图所示：

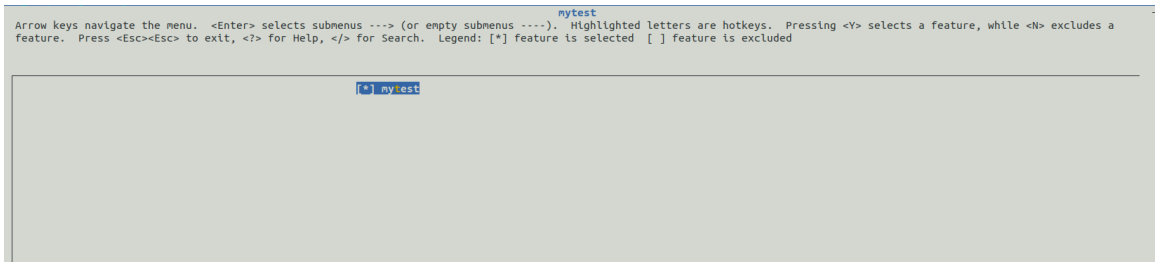



图 6.3.4.1 开启 mytest

配置完成后，我们就可以测试我们的 mytest 能不能编译了，命令如下所示：

```
make mytest
```

运行结果如下图所示：

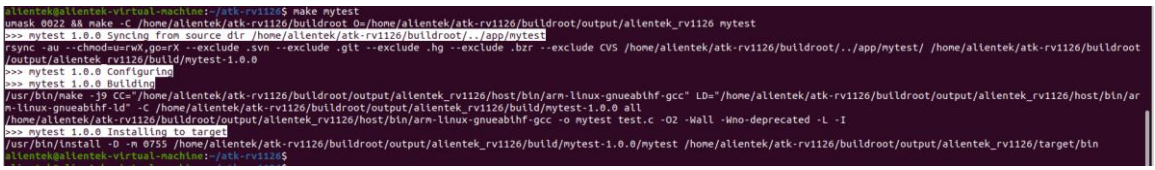



图 6.3.4.2 编译 mytest

## 6.4 最简单的添加开机自启

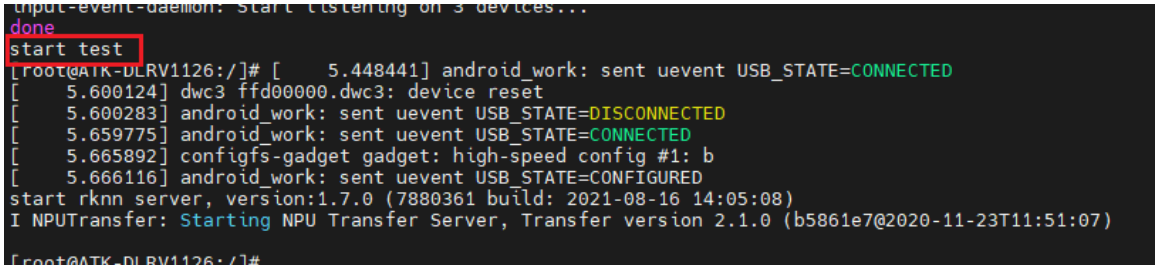
如何把自己的脚本添加到开发板上自启，烧录好文件系统后，在/etc/init.d/目录下就是开发板的自启脚本，创建一个脚本以“S”开头的 shell 脚本即可运行。比如我们创建一个 S99test 脚本，代码如下所示：

```
source /etc/profile
case "$1" in
  start)
        echo "start test"
        ;;
  stop)
        echo "stop test"
        ;;
  reload|force-reload)
        echo "Reloading test"
        "$0" reload
        ;;
  restart)
        "$0" stop
        sleep 1 # Prevent race condition: ensure QDesktop stops before
start.
        "$0" start
        ;;
  *)
        echo "Usage: $0 {start|stop|restart|reload|force-reload}"
esac
exit 0
```

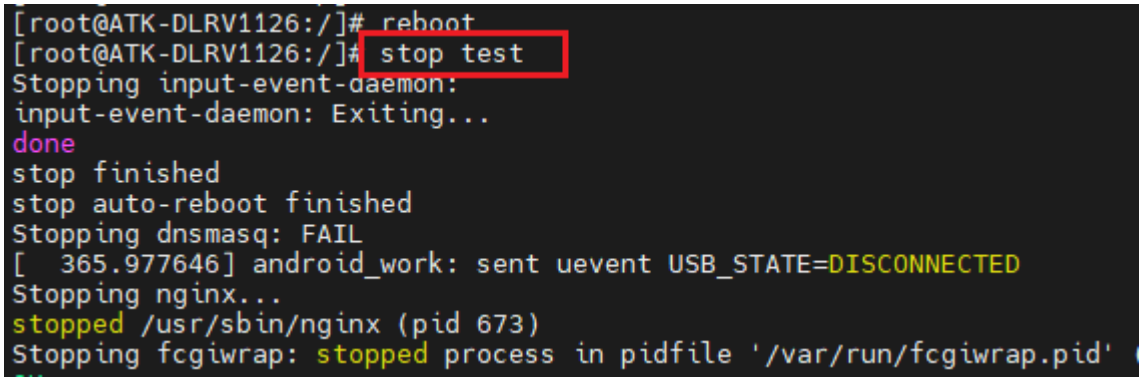
上述的代码很简单，不会的自己去看 shell case 使用。系统启动的时候，会调用这个脚本，传入的参数“start”就会运行打印“start test”。系统结束的时候，还是会调用这个脚本，传入的参数“stop”就会运行打印“stop test”。此脚本需要运行所以我们给个执行权限，命令如下所示：

```
chmod +x S99test
```

设置完成后，重启开发板即可看到打印信息：




6.4.1 系统启动打印




6.4.2 系统结束打印

可以把脚本放到 SDK 的镜像里，这样直接烧录就能启动自己的脚本，把脚本拷贝“buildroot/board/rockchip/rv1126_rv1109/fs-overlay-sysv/etc/init.d”目录下即可。例如：S99test 拷贝到这个目录下，结果如下图所示：

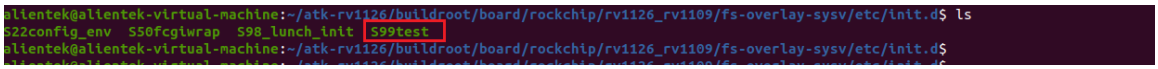



图 6.4.3 自启的脚本添加

添加完成后，重新编译 SDK 包，我们的脚本就能够自己启动了。

# 附录 A 使用技巧

## A1 摄像头的使用技巧

### A1.1 查看摄像头的初始化成功

如果想知道摄像头有么有正常的初始化，可以使用此命令进行测试：

```
dmesg | grep rockchip-mipi-dphy-rx
```

运行结果如下图所示：

```
[root@ATK-DLRV1126:/]# dmesg | grep rockchip-mipi-dphy-rx
[    0.721500] rockchip-mipi-dphy-rx ff4b0000.csi-dphy: match m00_f_imx335 1-001a-1:bus type 4
[    0.724480] rockchip-mipi-dphy-rx ff4b8000.csi-dphy: match m01_f_imx335 5-001a-1:bus type 4
[root@ATK-DLRV1126:/]#
[root@ATK-DLRV1126:/]#
```

这里笔者接了两个 imx335 摄像头，1-001a 表示 MIPI CSI0 初始化成功(靠近按键那个摄像头)，5-001a 表示 MIPI CSI1 初始化成功。

### A1.2 获取原始的 raw 数据

1. 获取 MIPI CSI0 的 raw 数据

首先我们需要摄像头能够正常工作，运行以下代码进行获取摄像头的数据(imx335)：

```
echo 0 > /sys/devices/platform/rkcif_mipi_lvds/compact_test
//设置数据以非紧凑型存储
v4l2-ctl -d /dev/video0 \
--set-fmt-video=width=2592,height=1944,\
pixelformat=GB10 \
--stream-mmap=3 \
--stream-skip=3 \
--stream-to=/tmp/cif.raw \
--stream-count=1 \
--stream-poll
```

第一条命令是设置数据以非紧凑型存储。第二条命令获取帧(video0 是接在 MIPI CSI0 上)

-d：指定摄像头的设备节点

--set-fmt-video：指定图像的分辨率，分辨率不能修改，修改后获取图像有问题。

--Pixelformat：指定文件格式，NV12。

--stream-mmap：指定 buffer 的类型为 mmap。

--stream-skip：指定丢弃的前 3 帧。

--stream-to：指定帧数保存的文件路径。

--stream-count：指定抓取的帧数

--stream-poll：v4l2-ctl 采用异步 IO。

注意：上述的命令是获取 IMX335 摄像头。下面的命令为 IMX415 摄像头：

```
echo 0 > /sys/devices/platform/rkcif_mipi_lvds/compact_test
//设置数据以非紧凑型存储
v4l2-ctl -d /dev/video0 \
--set-fmt-video=width=3840,height=2160,\
pixelformat= GB10 \
```

```
--stream-mmap=3 \
--stream-skip=3 \
--stream-to=/tmp/cif2.raw \
--stream-count=1 \
--stream-poll
```

获取后需要安装 7YUV 软件(版权问题就不提供下载，自己上百度下载)。拷贝 raw 文件到 Windows 系统下，用 7YUV 软件打开此文件，一开始是不能正常显示图片的，我们需要如下设置

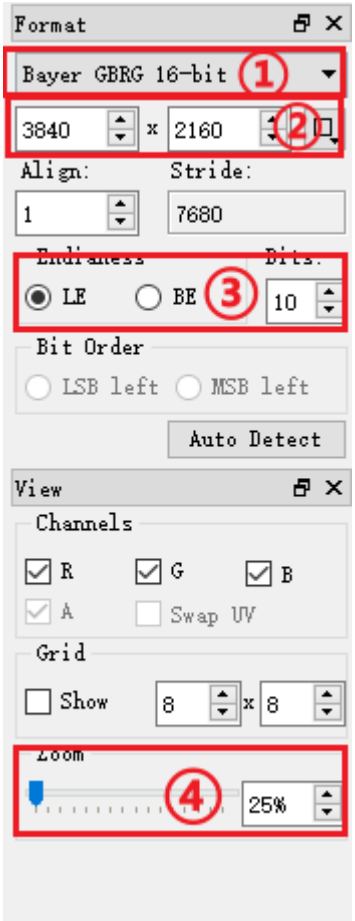



图 A1.2.1 设置 GB10 格式

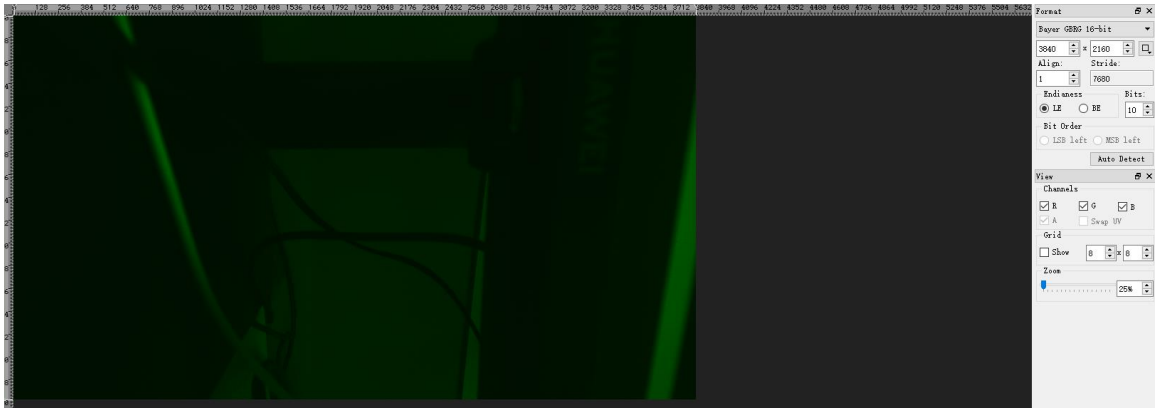
第一步：设置格式为 GBRG16 格式。

第二步：设置图片的大小，笔者是使用 IMX415 做测试，所以分辨率为 3840x2160。

第三步：设置数据为小端模式和 10Bit。GB10 表示一个像素只有 10Bit。

第四步：ZOOM 为图像的放大和缩小功能。

如果图像偏绿是正常的。打开图片结果如下图所示：



A1.2.27YUV 打开图片

2. 获取 MIPI CSI1 的 raw 数据

运行以下代码进行切换摄像头的数据

```
media-ctl -d /dev/media2-l '"rkisp-isp-subdev":2->"rkisp-bridge-ispp":0[0]'
media-ctl -d /dev/media2-l '"rkisp-isp-subdev":2->"rkisp-bridge-ispp":0[1]'
media-ctl -d /dev/media2 --set-v4l2 '"rkisp-isp-subdev":2 [fmt: SBGGR10_1X10/2592x1944]'
```

MIPI CSI1 像头的节点为 video17,例如 MIPI CSI1 接上 IMX335,所以命令修改为如下：

```
v4l2-ctl -d /dev/video17 \
--set-fmt-video=width=2592,height=1944,\
pixelformat=BG10 \
--stream-mmap=3 \
--stream-skip=3 \
--stream-to=/tmp/cif.raw \
--stream-count=1 \
--stream-poll
```

### A1.3 ISP 摄像头的节点

运行以下代码打印出摄像头的节点和别名

```
grep " /sys/class/video4linux/video*/name
```

运行结果如下图所示：

```
[root@ATK-DLRV1126:/]# grep '' /sys/class/video4linux/video*/name
/sys/class/video4linux/video0/name:stream_cif_mipi_id0
/sys/class/video4linux/video1/name:stream_cif_mipi_id1
/sys/class/video4linux/video10/name:rkisp_rawwr3
/sys/class/video4linux/video11/name:rkisp_rawrd0_m
/sys/class/video4linux/video12/name:rkisp_rawrd2_s
/sys/class/video4linux/video13/name:rkisp_rawrd1_l
/sys/class/video4linux/video14/name:rkisp-statistics
/sys/class/video4linux/video15/name:rkisp-input-params
/sys/class/video4linux/video16/name:rkisp-mipi-luma
/sys/class/video4linux/video17/name:rkisp_mainpath
/sys/class/video4linux/video18/name:rkisp_selfpath
/sys/class/video4linux/video19/name:rkisp_rawwr0
/sys/class/video4linux/video2/name:stream_cif_mipi_id2
/sys/class/video4linux/video20/name:rkisp_rawwr1
/sys/class/video4linux/video21/name:rkisp_rawwr2
/sys/class/video4linux/video22/name:rkisp_rawwr3
/sys/class/video4linux/video23/name:rkisp_rawrd0_m
/sys/class/video4linux/video24/name:rkisp_rawrd2_s
/sys/class/video4linux/video25/name:rkisp_rawrd1_l
/sys/class/video4linux/video26/name:rkisp-statistics
/sys/class/video4linux/video27/name:rkisp-input-params
/sys/class/video4linux/video28/name:rkisp-mipi-luma
/sys/class/video4linux/video29/name:rkispp_input_image
/sys/class/video4linux/video3/name:stream_cif_mipi_id3
/sys/class/video4linux/video30/name:rkispp_m_bypass
/sys/class/video4linux/video31/name:rkispp_scale0
/sys/class/video4linux/video32/name:rkispp_scale1
/sys/class/video4linux/video33/name:rkispp_scale2
/sys/class/video4linux/video34/name:rkispp_iqtool
/sys/class/video4linux/video35/name:rkispp_input_params
/sys/class/video4linux/video36/name:rkispp-stats
/sys/class/video4linux/video37/name:rkispp_input_image
/sys/class/video4linux/video38/name:rkispp_m_bypass
/sys/class/video4linux/video39/name:rkispp_scale0
/sys/class/video4linux/video4/name:rkcif-mipi-luma
/sys/class/video4linux/video40/name:rkispp_scale1
/sys/class/video4linux/video41/name:rkispp_scale2
/sys/class/video4linux/video42/name:rkispp_iqtool
/sys/class/video4linux/video43/name:rkispp_input_params
/sys/class/video4linux/video44/name:rkispp-stats
/sys/class/video4linux/video5/name:rkisp_mainpath
/sys/class/video4linux/video6/name:rkisp_selfpath
/sys/class/video4linux/video7/name:rkisp_rawwr0
/sys/class/video4linux/video8/name:rkisp_rawwr1
/sys/class/video4linux/video9/name:rkisp_rawwr2
[root@ATK-DLRV1126:/]#
```

上图中总共有 44 个 video 的节点，因为我们的 MIPI 摄像头的数据为 RAW，RAW 格式是原始的摄像头数据我们是无法使用的，需要经过 RV1126 的 ISP 处理后输出 RGB 或者 YUV 数据。ISP 处理后的摄像头节点有:

| Entity name | 设备节点 | 最大的宽度 | 支持的输出格式 |
|---|---|---|---|
| rkispp_m_bypass | video30/38 | 不支持设置分辨率，不支持缩放 | NV12/NV16/YUYV/FBC0/FBC2 |
| rkispp_scale0 | video31/39 | max width: 3264，最大支持 8 倍缩放 | NV12/NV16/YUYV |
| rkispp_scale1 | video32/40 | max width: 1280，最大支持 8 倍缩放 | NV12/NV16/YUYV |
| rkispp_scale2 | video33/41 | max width: 1280，最大支持 8 倍缩放 | NV12/NV16/YUYV |

每一个 MIPI 摄像头经过 ISP 处理后有 4 个设备节点可以调用，video30~33 为 MIPI CSI0，video38~41 为 MIPI CSI1。

### A1.4 IPC 的使用

在出厂系统下，首先退出 QT 的 UI 界面，开启 ISP 功能。命令如下所示：

```
ispserver &
```

可以使用“startup_app_ipc”命令去开启 IPC 功能，需要学习一下此命令的参数，

| 选项 | 描述 | 默认值 | 备注 |
|---|---|---|---|
| -I | ISP 下的摄像头切换 | 0 | 0：MIPI CSI0<br>1: MIPI CSI1 |
| -h | 显示的高 | 1280 | 无 |
| -w | 显示的宽 | 720 | 无 |
| -d | 设备节点 | rkispp_scale2 | rkispp_scale0<br>rkispp_scale1<br>rkispp_scale2 |
| -W | 输入的宽 | 1280 | 无 |
| -H | 输入的高 | 720 | 无 |
| -?/--help | 显示帮助信息 | 无 | 无 |

根据自己的硬件，决定命令的参数，这里笔者是接了 720 屏幕和 CSI0 接上了摄像头，所以命令如下所示：

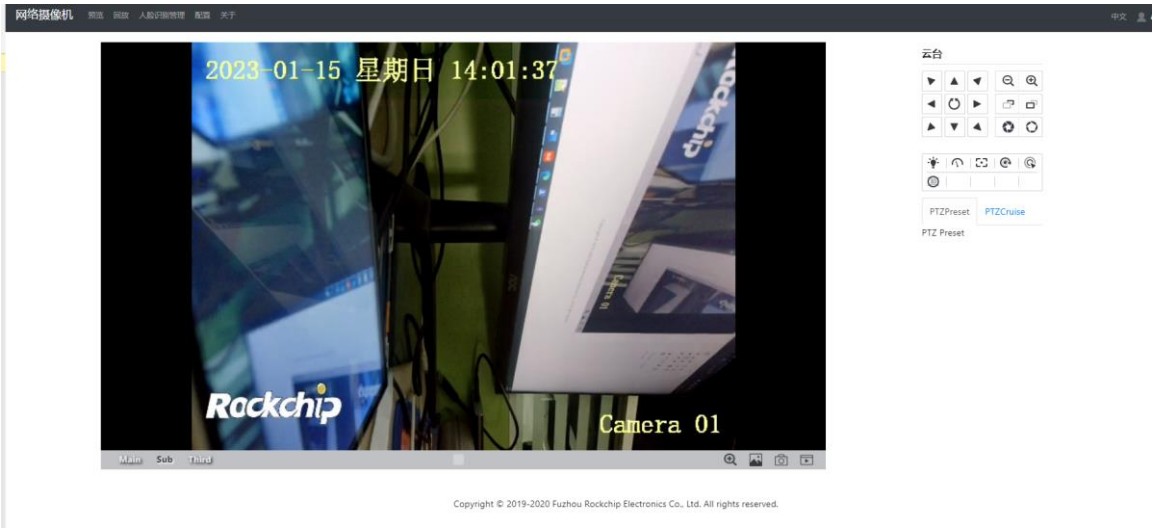
```
startup_app_ipc -I 0 &
```

这条命令开启 ipc，还是不能通过远程访问摄像头数据，还需要使用下面命令开启远程访问，命令如下所示：

```
RkLunch.sh
```

连接网络，使用“ifconfig”获取 IP 地址，笔者的 IP 地址为 192.168.6.184，打开浏览器输入开发板的 IP 地址，就会弹出以下窗口


用户名和密码都是：admin 登录就会显示摄像头的数据。


网络摄像机
2023-01-15 星期日 14:01:37
Rockchip
Camera 01
Main Sub Third
云台
PTZPreset
PTZCruise
PTZ Preset
Copyright © 2019-2020 Fuzhou Rockchip Electronics Co., Ltd. All rights reserved.

## A2 去掉正点原子的 QT 界面和 demo 分区

去掉 QT 界面和 demo 分区，可以按照以下步骤：

- 修改 buildroot 的配置

  修改正点原子的默认配置文件，在 SDK 的源码目录下“buildroot/configs/rockchip/alientek.config”，打开此文件去掉如下配置：

```
1   BR2_PACKAGE_ALIENTEK=y
2   BR2_PACKAGE_ATK_RKMEDIA=y
3   BR2_PACKAGE_ATK_RKMEDIA_LIBRARY=y
4   BR2_PACKAGE_ATK_DEMO=y
5   BR2_PACKAGE_QDESKTOP=y
61  BR2_PACKAGE_ATK=y
74  BR2_PACKAGE_ALIENTEK_DEMO=y
```

  在上面的每一行的首字母前，加“#”行即可注释掉。如下所示：

```
#BR2_PACKAGE_ALIENTEK=y
#BR2_PACKAGE_ATK_RKMEDIA=y
#BR2_PACKAGE_ATK_RKMEDIA_LIBRARY=y
#BR2_PACKAGE_ATK_DEMO=y
#BR2_PACKAGE_QDESKTOP=y
BR2_TARGET_GENERIC_HOSTNAME="ATK-DLRV1126"
BR2_TARGET_GENERIC_ISSUE="Welcome to ATK-DLRV1126 Buildroot"
BR2_TARGET_GENERIC_ROOT_PASSWD="root"
BR2_PACKAGE_CAMERA_ENGINE_RKAIQ_IQFILE="imx415_YT10092_IR0147-60IRC-8M-F20-hdr3.xml imx335_MTV4-IR-E-P_40IRC-4MP-F16.xml"
```

图 A2.1 buildroot 配置

- 修改 mkfirmware.sh 文件

  mkfirmware.sh 文件负责打包（此文件在源码的 SDK 目录下），所以我们需要去掉 demo 分区相关的配置。打开此文件修改注释掉 26 行、28 行和 52 行(在行首加“#”号即可注释)，如下图所示：

```
fi
#ATK_DEMO_FS_TYPE=ext4
USER_DATA_DIR=$TOP_DIR/device/rockchip/userdata/$RK_USERDATA_DIR
#DEMO_DIR=$TOP_DIR/buildroot/output/$RK_CFG_BUILDROOT/demo
MISC_IMG=$TOP_DIR/device/rockchip/rockimg/$RK_MISC
ROOTFS_IMG=$TOP_DIR/$RK_ROOTFS_IMG
ROOTFS_IMG_SOURCE=$TOP_DIR/buildroot/output/$RK_CFG_BUILDROOT/images/rootfs.$RK_ROOTFS_TYPE
RAMBOOT_IMG=$TOP_DIR/buildroot/output/$RK_CFG_RAMBOOT/images/ramboot.img
RECOVERY_IMG=$TOP_DIR/buildroot/output/$RK_CFG_RECOVERY/images/recovery.img
if which fakeroot; then
FAKEROOT_TOOL="`which fakeroot`"
else
    echo -e "Install fakeroot First."
    echo -e "  sudo apt-get install fakeroot"
    exit -1
fi
OEM_FAKEROOT_SCRIPT=$ROCKDEV/oem.fs
USERDATA_FAKEROOT_SCRIPT=$ROCKDEV/userdata.fs
TRUST_IMG=$TOP_DIR/u-boot/trust.img
UBOOT_IMG=$TOP_DIR/u-boot/uboot.img
BOOT_IMG=$TOP_DIR/kernel/$RK_BOOT_IMG
LOADER=$TOP_DIR/u-boot/*_loader_v*.bin
SPL=$TOP_DIR/u-boot/*_loader_spl.bin
#SPINOR_LOADER=$TOP_DIR/u-boot/*_loader_spinor_v*.bin
MKIMAGE=$SCRIPT_DIR/mk-image.sh
mkdir -p $ROCKDEV

#$MKIMAGE $DEMO_DIR $ROCKDEV/demo.img $ATK_DEMO_FS_TYPE
# Require buildroot host tools to do image packing.
```

图 A2.2 修改 mkfirmware.sh 文件

在源码目录下，打开此文件“tools/linux/Linux_Pack_Firmware/rockdev/rv1126_rv1109-package-file”，注释掉第 24 行，如下图所示：

```
#resource   Image/resource.img
#kernel     Image/kernel.img
boot        Image/boot.img
recovery    Image/recovery.img
rootfs      Image/rootfs.img
oem     Image/oem.img
userdata    Image/userdata.img
#demo    Image/demo.img
```

图 A2.3 注释 demo 打包路径

- 修改 parameter-buildroot-fit.txt 文件

  此文件主要是负责上位机如何给 emmc 分区或者 SD 卡(路径：device/rockchip/rv1126_rv1109/parameter-buildroot-fit.txt)。所以我们需要去掉最后的 demo 分区，修改第 11 行，删除 demo 分区如下：

(userdata),0x00200000@0x00498000(media), -@0x00698000(demo:grow) #原本的配置
(userdata),-@0x00498000(media:grow) #删除掉 demo 分区

  如下图所示：

```
TYPE: GPT
CMDLINE: mtdparts=rk29xxnand:0x00002000@0x00004000(uboot),0x00002000@0x00006000(misc),0x00010000@0x00008000(boot),0x00010000@0x00018000(recovery),0x00010000@0x00028000(backup),0
x00200000@0x00038000(rootfs),0x00060000@0x00238000(oem),0x00200000@0x00298000(userdata),-@0x00498000(media:grow)
uuid:rootfs=614e0000-0000-4b53-8000-1d28000054a9
```

图 A2.4 修改 parameter-buildroot-fit.txt 文件

- 开启瑞芯微的 IPC 程序

  如果需要开机默认开启官方的 IPC 程序(正点原子默认是关闭的)，需要做以下配置修改：

  修改 S98_lunch_init 文件(此文件在 SDK 源码目录下：buildroot/board/rockchip/rv1126_rv1109/fs-overlay-sysv/etc/init.d/S98_lunch_init)，把此文件的每一行的首个“#”行去掉，去掉完成如下图所示：

```
source /etc/profile.d/RkEnv.sh

case "$1" in
    start)
        [ -f /oem/RkLunch.sh ] && source /oem/RkLunch.sh
        #recovery test
        if [ -e "/oem/rockchip_test/auto_reboot.sh" ]; then
            mkdir /data/cfg/rockchip_test
            cp /oem/rockchip_test/auto_reboot.sh /data/cfg/rockchip_test
            source /data/cfg/rockchip_test/auto_reboot.sh &
        fi
        ;;
    stop)
        [ -f /oem/RkLunch-stop.sh ] && source /oem/RkLunch-stop.sh
        printf "stop finished\n"
        ;;
    *)
        echo "Usage: $0 {start|stop}"
        exit 1
        ;;
esac
exit 0
```

图 A2.5 开启 S98_lunch_init

  修改 startup_app_ipc.mk 文件，文件的目录为：buildroot/package/rockchip/startup_app_ipc/startup_app_ipc.mk，打开此文件开启 13~18 行的配置(去掉“#”行即可)，如下图所示：

```
STARTUP_APP_IPC_INIT_SCRIPT=package/rockchip/startup_app_ipc/S04startup_app_ipc

define STARTUP_APP_IPC_INSTALL_INIT_SYSV
$(INSTALL) -D -m 0755 ${STARTUP_APP_IPC_INIT_SCRIPT} \
                $(TARGET_DIR)/etc/init.d/S04startup_app_ipc
endef

$(eval $(cmake-package))
```

图 A2.6 开启 S04startup_app_ipc