飞腾派 软件开发手册

版本	日期	更改记录	批准
0.5	2023.04.04	基于 V2 硬件开始编写	
0.9	2023.06.06	生成初版文档	
1.0	2023.08.03	形成正式版文档	

深圳中电港技术股份有限公司·萤火工场
广东省深圳市前海深港合作区南山街道自贸西街 151 号招商局前海经贸中心一期 A 座 20 层

目录

前言

本文档用于指导用户的相关应用和开发工作，它是用户设计与之相关的软硬件系统的官方参考文档。

本用户手册适用于具有一定技术能力的开发者。

免责声明

本文档仅提供阶段性数据，并不保证该数据的准确性及完整性。深圳中电港技术股份有限公司和飞腾信息技术有限公司共同对此文档内容享有最终解释权，且保留随时更新、补充和修订的权利。

本开源硬件和软件，及其他相关文档资料仅用于指导用户学习，不得用于商用用途。如果用户用于商用，所带来的一切风险和损失，由用户自行承担。

如有技术问题，可通过 support@cecport.com 获取支持，因不当使用本文档造成的损失，概不承担任何责任。

商标声明

在本用户手册出现或提及的其他非中电港、飞腾产品名称、徽标、品牌和其他商标均为财产各自的商标持有人，这些商标持有人不隶属于中电港、飞腾。

版权声明

技术支持

用户可以通过 Web，E-mail 等方式获得技术支持：

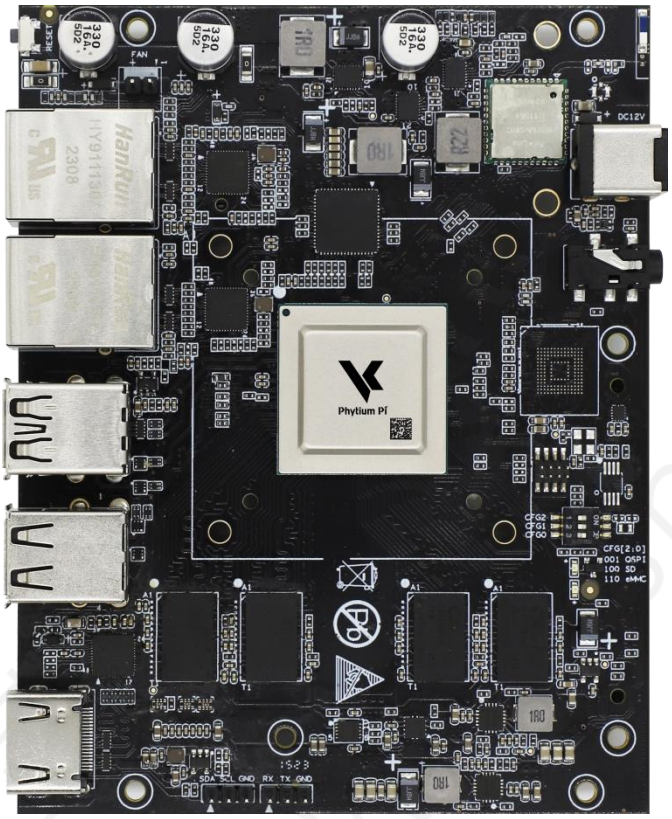
Web：www.cecport.com

E-mail：support@cecport.com

产品介绍篇

1．初识飞腾派

1.1．飞腾派简介



飞腾派开发板是萤火工场研发的一款面向行业工程师、学生和爱好者的开源硬件。

主板处理器采用飞腾定制四核处理器，该处理器兼容 ARM V8 指令集，包含 2 个 FTC664 核和 2 个 FTC310 核，其中 FTC664 核主频可达 1.8GHz，FTC310 核主频可达 1.5GHz。主板板载 64 位 DDR4 内存，有 2G 和 4G 两个版本，支持 SD 或者 eMMC 外部存储。主板板载 WiFi 和蓝牙，陶瓷天线，可快速连接无线通信。另外还集成了大量外设接口，包括双路千兆以太网、USB、UART、CAN、HDMI、Audio 等接口，集成一路 miniPCIE 接口，可实现 AI 加速卡与 4G、5G 通信等多种功能模块的扩展。

主板操作系统支持 Ubuntu、Debian 等国外主流开源操作系统，也支持国内 OpenKylin、OpenHarmony、SylixOS、RT-Thread 等国产操作系统。

1.2. 硬件规格

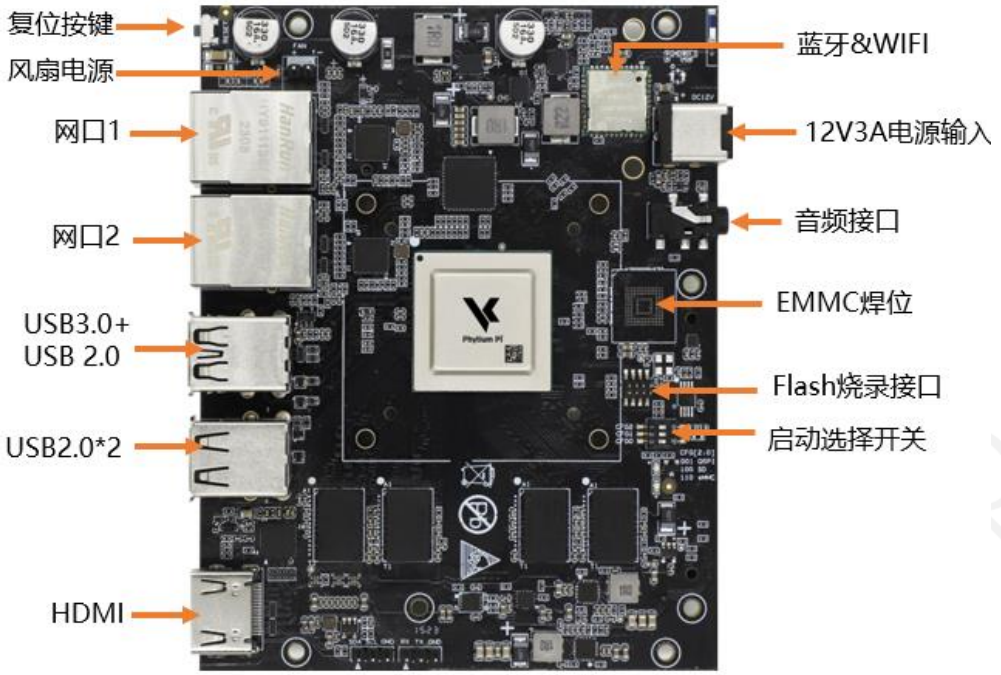
功能	描述
CPU	飞腾四核处理器，ARMV8 架构，2×FTC664@1.8GHz+2×FTC310@1.5GHz
内存	64 位 DDR4，分 2G 版本和 4G 版本
存储	支持 microSD 和 EMMC 启动，二选一
网络	2×千兆以太网（RJ45）
USB	1×USB3.0 host，3×USB2.0 host
PCIe	1×Mini-PCIe，支持 AI、5G\4G 等模组
蓝牙	板载蓝牙 BT4.2/BLE4.2
WiFi	板载 2.4G + 5G 双频 WIFI
4G/5G	可通过 miniPCIE 扩展实现
AI 加速	可通过 miniPCIE 扩展实现
显示	1×HDMI，最高支持 1920*1080 分辨率
视频解码	2K30p(H.264/265)|1080p60
音频	3.5mm 耳机口音频输出
UART	1×调试串口+2×MIO（多功能 IO，可配置为 UART 模式）
I2C	2+2×MIO（多功能 IO，可配置为 I2C 模式）
I2S	1 路
SPI	2 路
CAN	1 路 CANFD
GPIO	最多 29 个
SIM 卡	支持 1 路 SIM 卡
SD 卡	支持 1 路 SD 卡
LED 灯	电源指示灯和状态指示灯
供电要求	12V3A 直流电源
工作温度	0~50℃
产品尺寸	119mm×93mm
产品重量	72±2g
工作温度	0~50℃

1.3. 默认软件列表

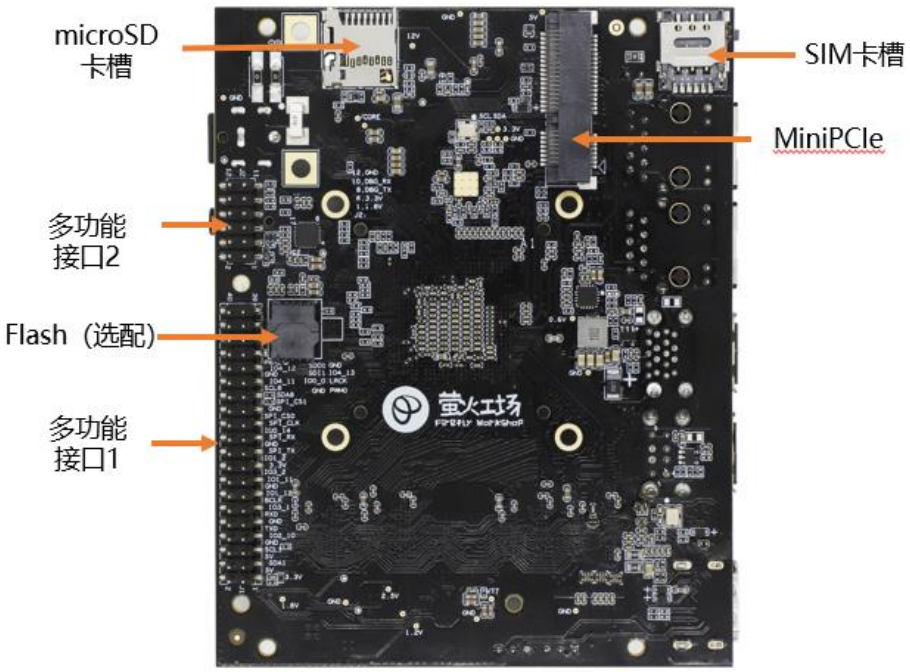
类别	名称	描述	支持情况
Bootloader	U-boot	基于 2022.01 版 uboot 定制	Yes
Linux 内核	Linux Kernel	基于 4.19.246 版 Linux Kernel 定制	Yes
外设驱动	USB Host	USB Host 驱动	Yes
	I2C	I2C 总线驱动	Yes
	SPI	SPI 总线驱动	Yes
	Ethernet	YT8521 驱动	Yes
	eMMC/SDIO	eMMC/SDIO 驱动	Yes
	PWM	PWM 驱动	Yes
	ADC	ADC 驱动	Yes
	GPIO	GPIO 驱动	Yes
	RTC	DS1339 驱动	Yes
	UART	UART 驱动	Yes
	CAN	CAN 驱动	Yes
	WiFi	RTL8821CS 驱动	Yes
	Bluetooth	RTL8821CS 驱动	Yes
	DP	DP 驱动（外接显示接口为 HDMI，由专用 DP 转 HDMI 芯片实现转换）	Yes
	I2S	I2S 驱动 codec ES8336 驱动	Yes
	VPU	(后续文件系统提供支持)	Yes
文件系统	基于 Ubuntu20.04 arm64 版本定制	xface 桌面，集成 Qt/OpenCV 开发环境	Yes
	Qt	5.12.8	Yes
	Python	3.8.10	Yes
	OpenCV	4.2.0	Yes

1.4．功能接口

顶层接口视图及说明：



底层接口视图及说明：

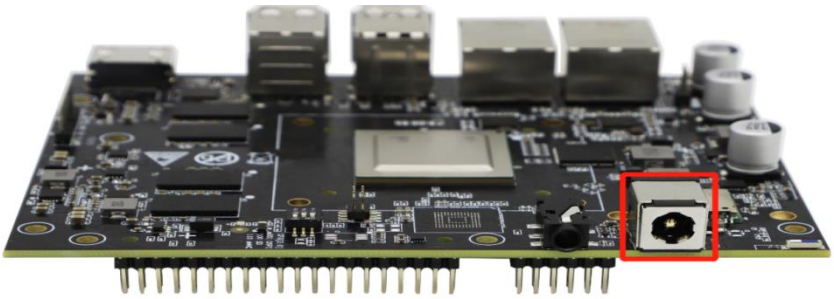


说明：以上照片系选取我司某一批次生产的板卡进行拍摄，由于产品在不断维护，可能实际出货的板卡可能与照片不尽一致。

1.5．接线与安装

1.5.1．电源

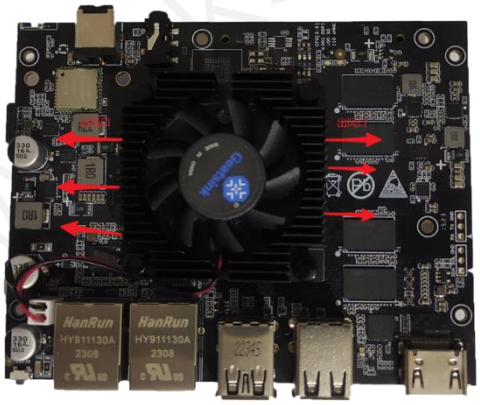
需使用 12V3A 直流电源供电，接口孔径 5.5mm，内径 2.1mm，内正外负，一头插入板子电源接口，另一头插电源插座上。



1.5.2．风扇

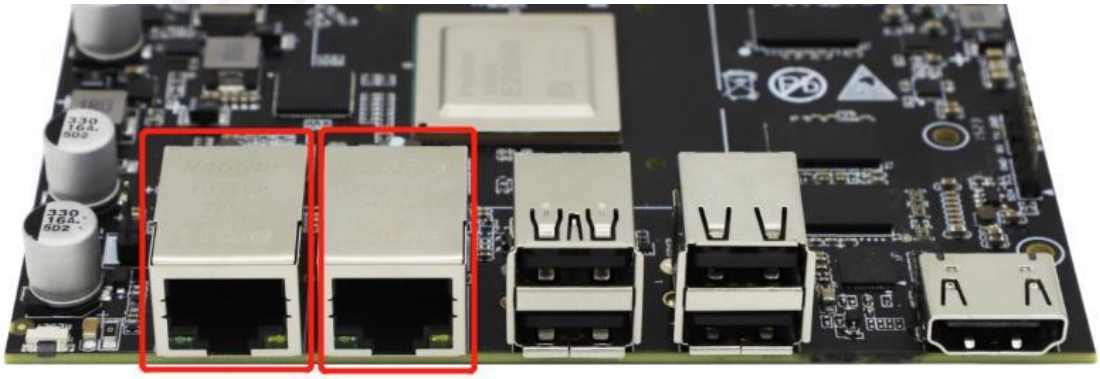
待撕除风扇背胶后，根据固定孔位按压风扇弹簧固定柱对风扇进行固定，最后连接风扇电源线。风扇电源的正负极（红正黑负）需与板上丝印匹配（FAN+、FAN-），请依据板上正负极标识连接风扇电源线，切勿接反。

风扇安装注意出风的两侧方向朝向电源电路和 DDR，以获得最佳的散热效果。



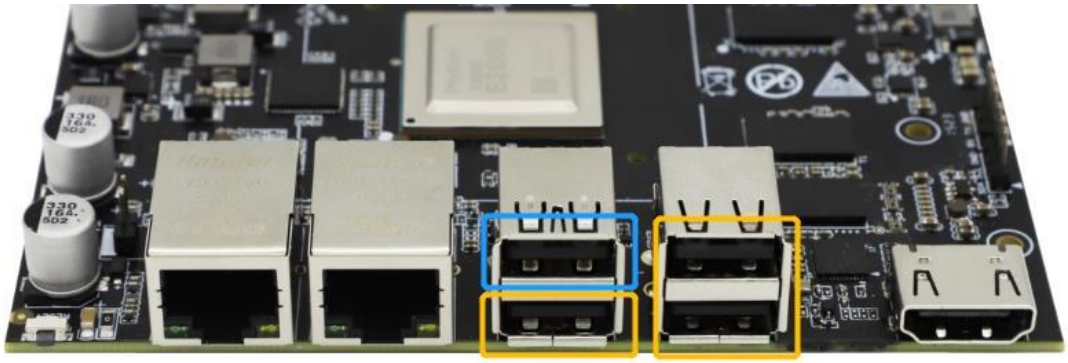
1.5.3．网口

板载 2 路千兆以太网接口，左 1 为 eth1，左 2 为 eth0。



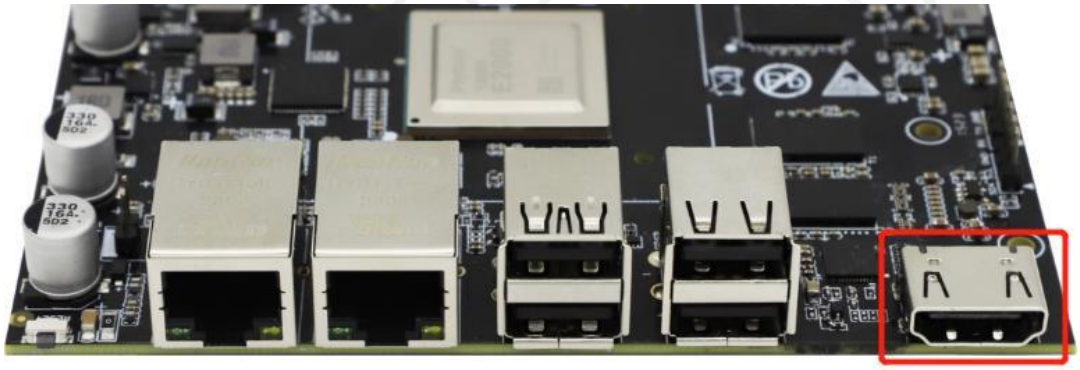
1.5.4. USB

板载 4 路 USB，其中 1 路 USB3.0（蓝色方框）和 3 个 USB2.0（黄色方框），建议 USB2.0 连接键鼠，USB3.0 用于连接外设，以利用最佳性能。



1.5.5. HDMI

支持一路 HDMI 显示，最大分辨率 1920*1080。HDMI 是由 DP 经过转换芯片 LT8711UXD 转换后所得。



基础操作篇

2．系统安装

2.1．系统简介

飞腾派支持支持 Ubuntu、Debian 等国外主流开源操作系统，也支持国内 OpenKylin、OpenHarmony、SylixOS、RT-Thread 等国产操作系统，飞腾派将陆续适配更多有影响力的操作系统和开源软件。

本手册基于 Ubuntu 系统编写，以下内容均在 Ubuntu 系统下实现，其他操作系统请联系对应销售人员获取技术支持。

2.2．系统安装与恢复

飞腾派标准板默认完全从 SD 卡启动系统，如需从 FLASH 或 EMMC 启动，请联系销售人员获取支持。

首先需要将启动拨码开关配置成 SD 卡启动模式，配置方法如下图：

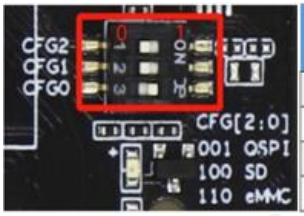


表 4 启动选择拨码开关配置表

CFG2	CFG1	CFG0	引导启动方式
0	0	1	从 QSPI FLASH 引导启动
1	0	0	从 SD 卡引导启动
1	1	0	从 EMMC 引导启动

2.2.1．镜像获取

（1）获取镜像

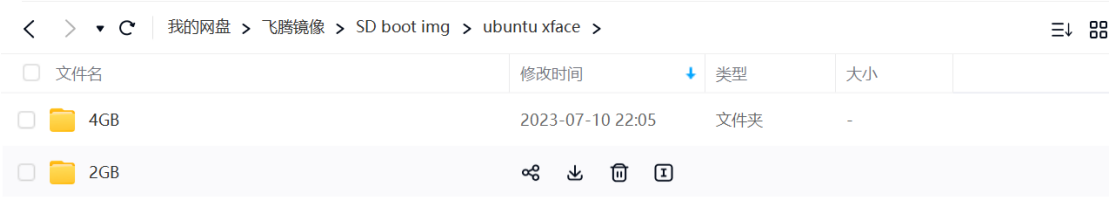
通过如下百度云盘链接或者联系销售获取到镜像，镜像链接：

https://pan.baidu.com/s/17UV4A3PDzaix9xPeK-gyRg　　提取码：6hur

注：通过百度网盘获取镜像和烧录工具（网盘会不定期更新，具体查看实际查看到的信息），如下图：

（2）下载对应版本

镜像需下载对应内存大小的版本，目前 4GB DDR 使用长鑫存储（CXMT）的颗粒，2GB DDR 使用兆易创新（GigaDevice）的颗粒，如果不清楚开发板是哪个容量的 DDR，可通过 DDR 颗粒的上丝印字样来判断。



（3）查看校验工具

提供的工具有两个，分别用于镜像校验和镜像烧录：

2.2.2. 安装准备

（1）准备工作

准备一张存储空间 16G 以上的 SD 卡、一个 USB 读卡器、一个 USB 转 TTL 串口调试器（建议准备）：

（MicroSD 卡）

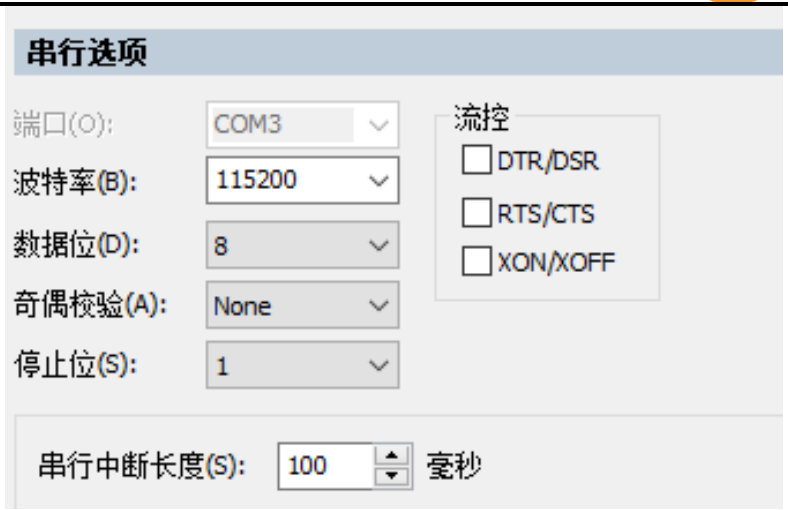
（USB 读卡器）

（USB 转 TTL 串口调试器）

（2）连接电脑

将 USB 转 TTL 模块的 USB 接口一端插入到电脑的 USB 接口中。（为了更好的兼容性，推荐使用 CH340 USB 转 TTL 模块。购买模块前，请确认模块支持 115200 速率的波特率。）

以下图示我们串口接到 12PIN 对应引脚上，串口工具波特率选择 115200，数据位 8，奇偶校验 None，停止位 1：



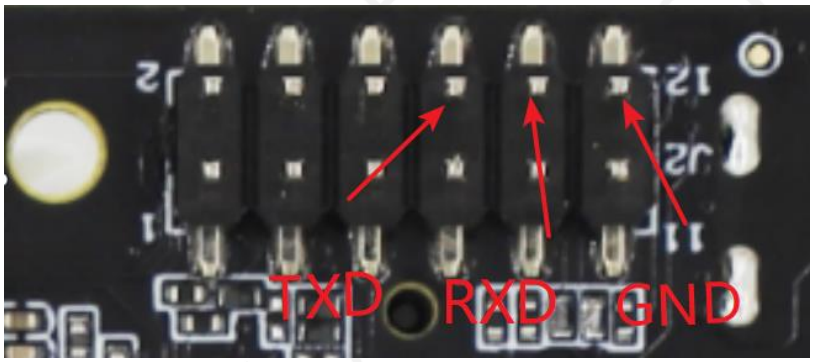
（3）连接开发板

USB 转串口调试器的引脚需要通过杜邦线连接到开发板的调试串口上，注意收发交叉连接：

a.USB 转 TTL 模块的 GND 接到开发板的 GND（12 脚）上

b.USB 转 TTL 模块的 RX 接到开发板的 TX 上（8 脚）

c.USB 转 TTL 模块的 TX 接到开发板的 RX 上（10 脚）



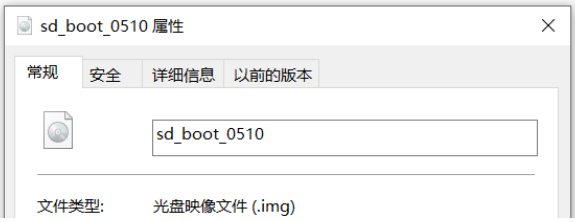
2.2.3. 镜像烧录

飞腾派通过配置开发板上的拨码开关，可实现不同的启动方式。

以下为 SD 卡启动的烧录方式步骤：

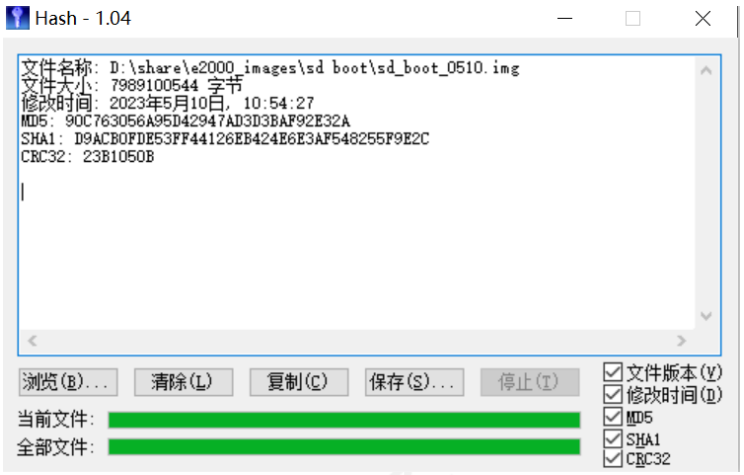
（1）解压镜像

下载好镜像后用解压工具解压出格式为 img 的镜像文件如下图：



（2）校验镜像

校对镜像的 checksum，对比网盘提供的 checksum，必须一致：

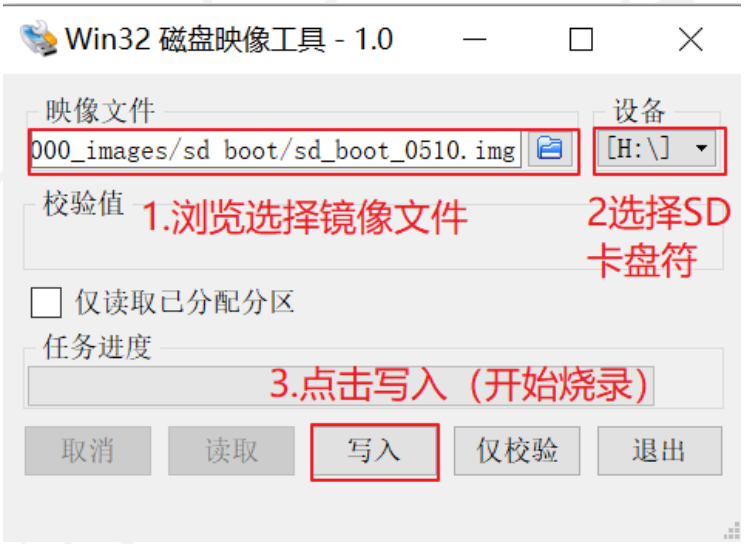


（3）安装软件

安装烧录软件 Win32DiskImager（检查是否已下载烧录软件，若无该软件，请重新下载）。

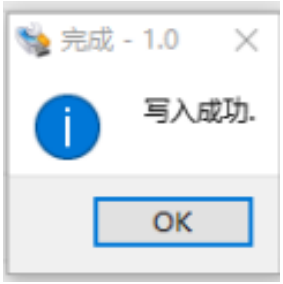
（4）开始烧录

打开 Win32DiskImager 软件，并按照如下提示开始烧录：



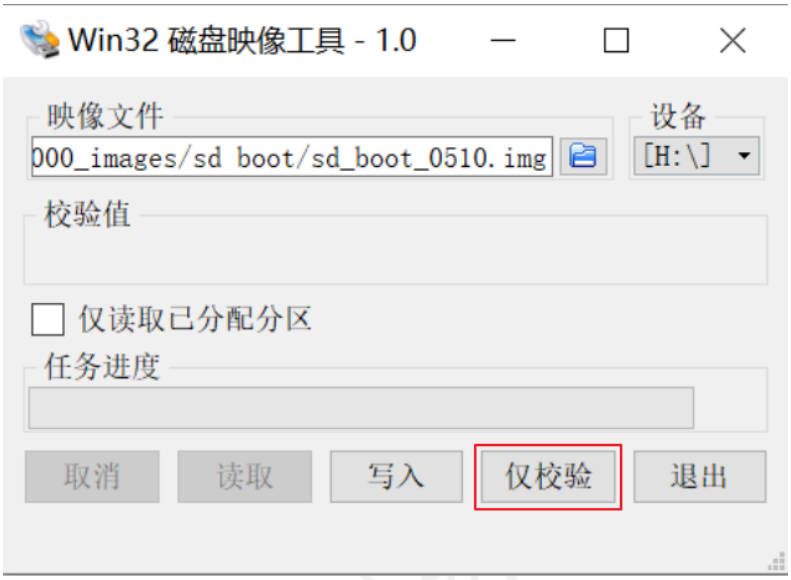
（5）烧录完成

查看任务进度，达到 100%后，会弹出写入成功提示“写入成功”。



（6）校验烧录

烧录完点击“仅校验”确认烧录正确，若校验成功，可进行下一步。



（7）安装 SD 卡

烧录完成后安装到 SD 卡槽即可上电启动，注意要将启动模式设置为 SD 卡启动。

注：注意区分 2G 版与 4G 版镜像。

2.2.4．Linux 环境下的烧录镜像

若是在 linux 环境下烧录新的 SD 卡,可以直接进行以下操作(如果镜像是 xxx.img.xz 格式，需要先解压成 img 格式，再使用 dd 烧写)：

（1）通过 dd 命令进行安装镜像

```
dd if=sd_boot_xface_2GB_0625_mini.img of=/dev/sdx
```

if=后面的是要安装的镜像的名称（注意路径），of=后面的是你的 SD 卡或 U 盘，sdX 的 X 需要改为存储卡实际映射值，如 sda，sdb 等（通过 lsblk 命令查看）。

注意：在烧录之前，最好先将 sd 卡的挂载点卸载掉，以防有其他程序读写。比如：

```
umount /dev/sda1
umount /dev/sda2
```

2.3．启动方式设置

启动选择开关，开发板可通过手动（可以用镊子、螺丝刀等）拨动拨码开关来配置启动方式，具体配置如下：

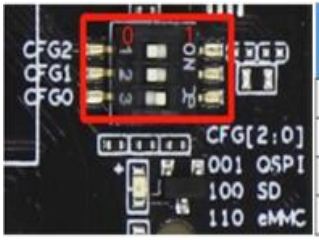


表 4 启动选择拨码开关配置表

CFG2	CFG1	CFG0	引导启动方式
0	0	1	从 QSPI FLASH 引导启动
1	0	0	从 SD 卡引导启动
1	1	0	从 EMMC 引导启动

拨码	uboot	kernel	rootfs
001	flash	EMMC	EMMC
001	flash	microSD	microSD
100	microSD	microSD	microSD
110	EMMC	EMMC	EMMC

2.4．Linux 系统目录

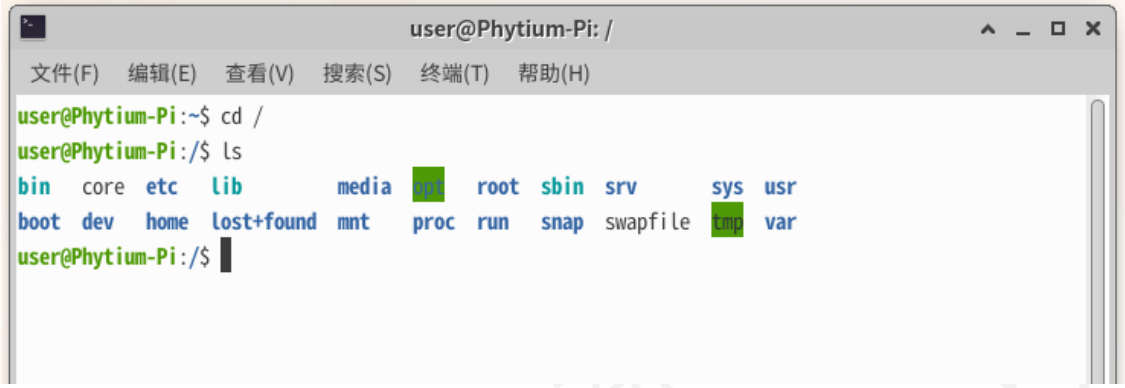
Linux 没有炫目的可视化操作界面，在 Linux 系统中，所有内容都是以文件的形式保存和管理的，即「一切皆文件」。普通文件是文件，目录（Windows 下称为文件夹）是文件，硬件设备（键盘、监视器、硬盘、打印机）是文件，就连套接字（socket）、网络通信等资源也都是文件。

linux 只有一个根目录，而且文件和目录被组织成一个单根倒置树结构，此结构最上层是根目录，用“/”表示根文件系统(rootfs)：root filesystem：

2.4.1．终端方式查看

启动飞腾派，按照之前的操作方式，打开飞腾派终端：

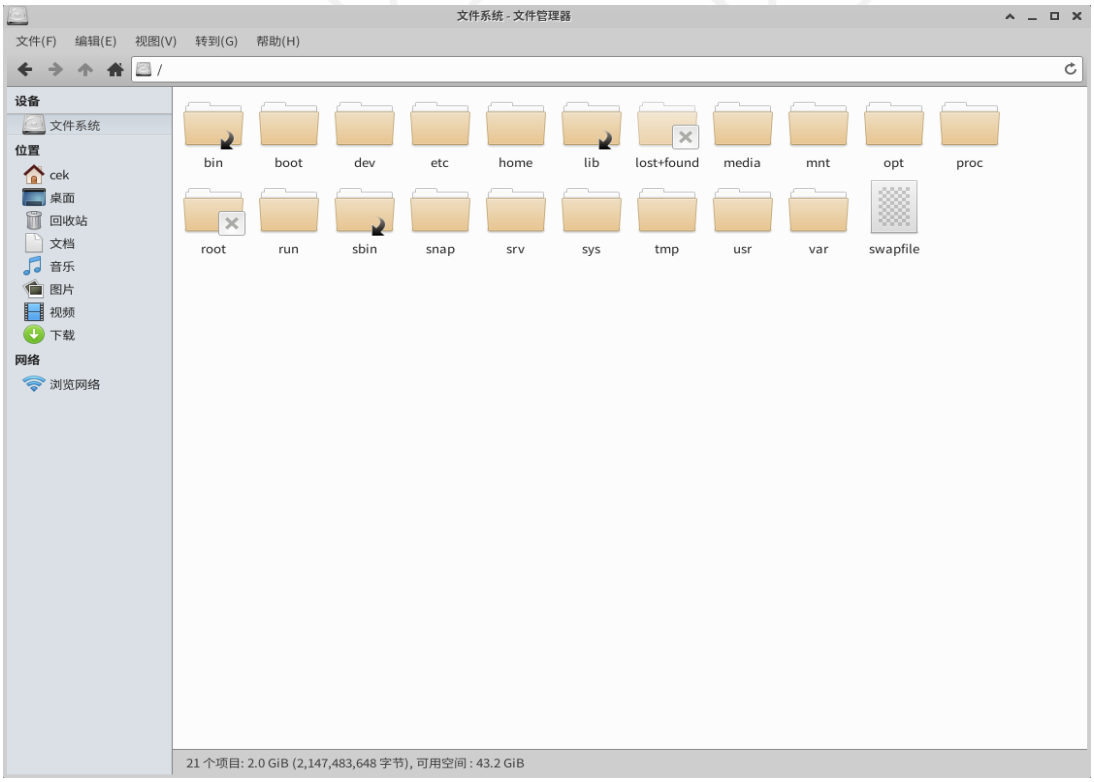
按照下图所示划红线部分，依次输入命令“cd /”+回车切换到根目录，和命令”ls”+回车浏览目录下文件（注意：cd 和..中间有一空格）：



2.4.2．文件浏览器方式查看

使用图形界面访问，可通过 HDMI 连接显示器后上电开机进入图形操作界面。

找到文件系统-文件管理器进行查看：



2.5．终端登录与文件传输

本节讲述的应用需要对飞腾派的 Wi-Fi 网络或有线网络进行配置，确保网络通畅之后通过 SSH 或 WinSCP 实现登陆。

2.5.1．SSH 登录

（1）安装 ssh 环境

```
$ sudo apt-get install openssh-server
```

（2）启动服务器的 SSH 服务

确认 ssh-server 是否已经启动：

```
$ ps -e | grep ssh
```

sshd 表示 ssh-server 已经启动了。如果没有启动，可以使用如下命令启动：

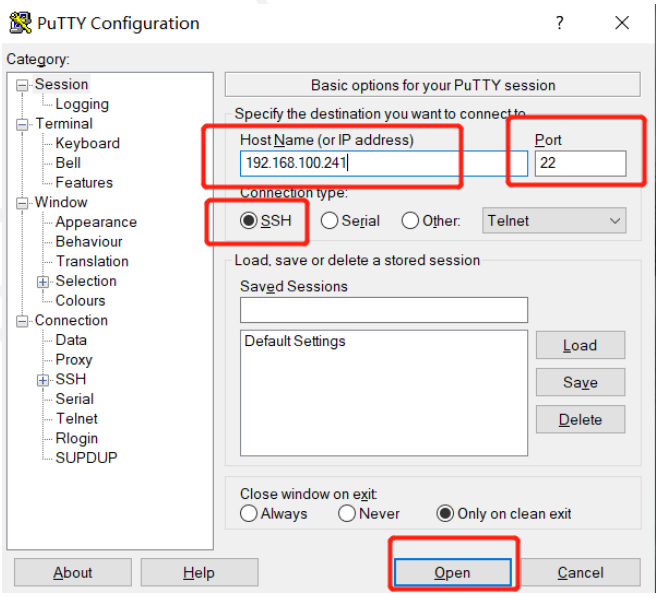
```
$ sudo /etc/init.d/ssh start
```

停止和重启 ssh 服务的命令如下：

```
$ sudo /etc/init.d/ssh stop    #server 停止 ssh 服务
$ sudo /etc/init.d/ssh restart    #server 重启 ssh 服务
```

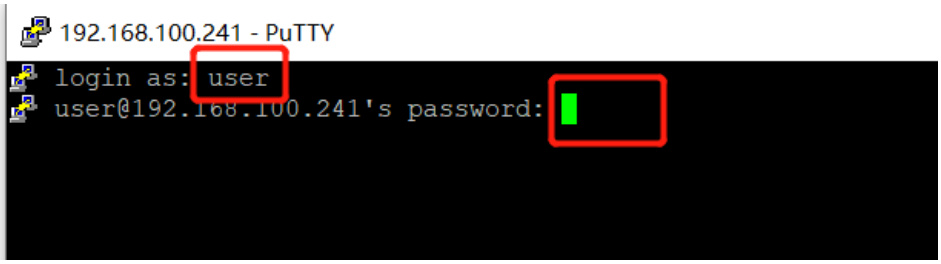
（3）快速建立连接

打开 PC 端的 SSH 登录软件，这里以 putty 为例，输入 IP “xxx.xxx.xxx.xxx”（输入 IP 根据用户需要的 IP 进行输入，用 ifconfig 显示 IP 地址，确保飞腾派和 PC 处于同一网络），输入用户名为 “user”，首次登陆会出现一个安全弹窗警告，点击 “接受并保存” 选项即可。



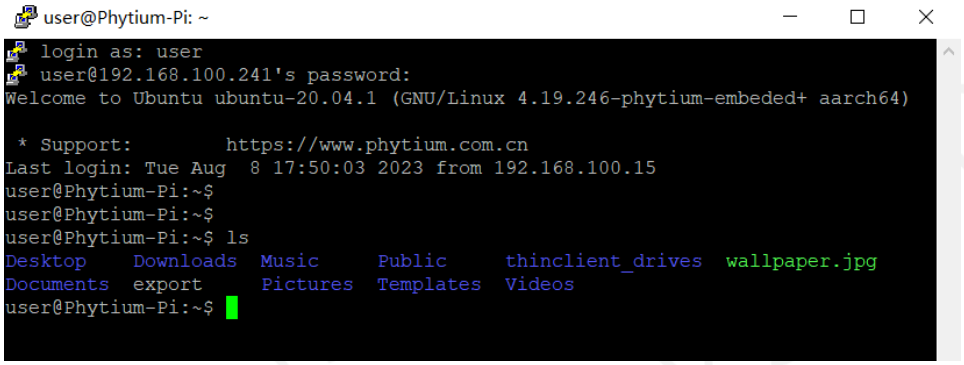
（4）输入登录密码

输入用户名：user ， 密码：user，密码输入不显示，点击“回车键”登录。



（5）登录成功

登录成功后，串口将显示以下信息：

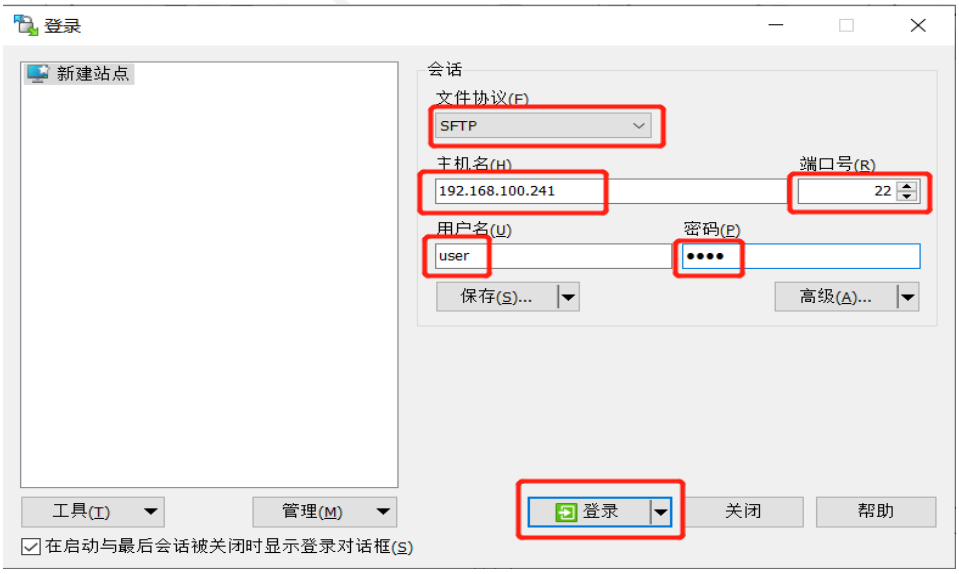


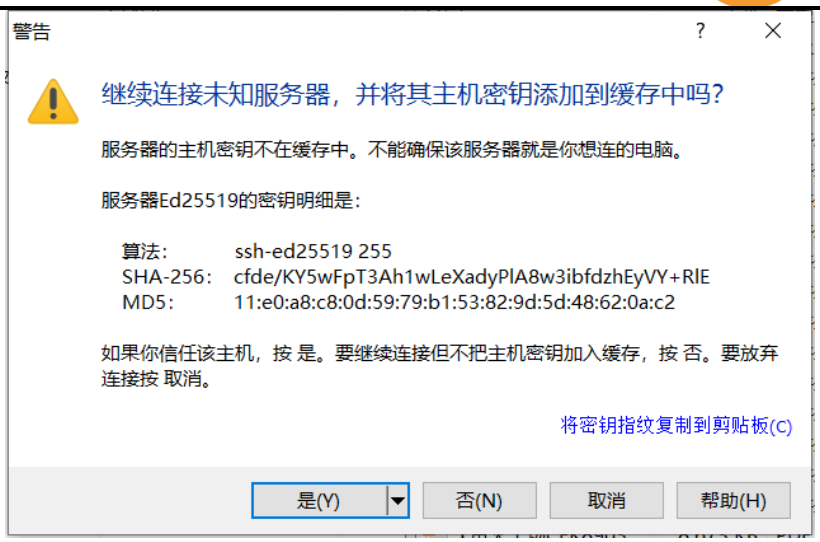
2.5.2. WinSCP

WinSCP 是一个 Windows 环境下使用 SSH 的开源图形化 SFTP 客户端，同时支持 SCP 协议。这里也可以使用其他 FTP 工具建立文件传输。

（1）新建站点

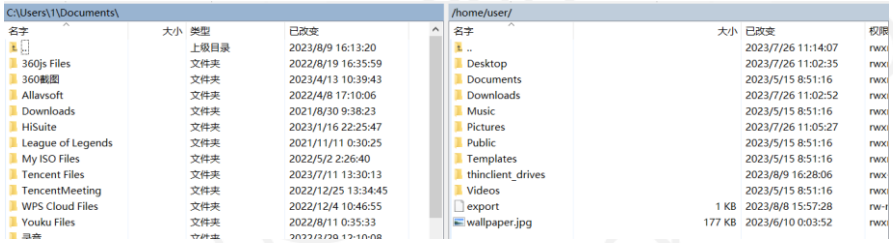
输入 IP “xxx.xxx.xxx.xxx” （输入 IP 根据用户需要的 IP 进行输入，用 ifconfig 显示 IP 地址，确保飞腾派和 PC 处于同一网络），输入用户名为“user”，密码为“user”。首次登陆会出现一个安全弹窗警告，点击“是” 选项即可。





（2）文件传输

上一步建立完成后，会显示文件目录，可以把需要的文件放置对应文件夹中。



2.6．XRDP 远程桌面

2.6.1．安装

板端使用下面的命令安装 xrdp：

```
$ sudo apt-get install xrdp -y
```

如图示，由于之前已经安装过，所以显示 XRDP 已经是最新版：

```
user@Phytium-Pi:~$ sudo apt-get install xrdp -y
[sudo] user 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
xrdp 已经是最新版 (0.9.12-1)。
下列软件包是自动安装的并且现在不需要了:
  appmenu-gtk-module-common apt-config-icons-hidpi apt-config-icons-large
  apt-config-icons-large-hidpi dconf-cli ffmpeg ffmpegthumbs freerdp2-x11
  geoip-database gnome-menus gnome-session-bin gnome-session-common
  gnome-startup-applications gnustep-base-common gnustep-base-runtime
  gnustep-common ibus-gtk ibus-gtk3 kamera kdegames-card-data-kf5
  kdegames-mahjongg-data-kf5 kpeople-vcard libappmenu-gtk3-parser0
  libavdevice58 libchm1 libdee-1.0-4 libdjvulibre-text libdjvulibre21
  libfakekey0 libfreecell-solver0 libfreerdp-client2-2 libfreerdp2-2 libgeoip1
  libgeonames-common libgeonames0 libgnome-menu-3-0 libgnustep-base1.26
  libkf5baloowidgets-bin libkf5baloowidgets-data libkf5baloowidgets5
  libkf5contacts-data libkf5contacts5 libkf5jsapi5 libkf5kdegames-data
  libkf5kdegames7 libkf5kdegamesprivate1 libkf5plotting5 libkf5pulseaudioqt2
  libkf5sane-data libkf5sane5 libkf5syndication5abi1 libkf5webkit5
  libmarkdown2 libqmobipocket2 libsnapd-qt1 libspectre1 libtimezonemap-data
  libtimezonemap1 libunity-control-center1 libvncclient1 libwinpr2-2
  libzeitgeist-2.0-0 p7zip p7zip-full plasma-discover-common python3-ibus-1.0
  python3-pyxattr qml-module-org-kde-people qtspeech5-flite-plugin rtmpdump
  sshfs ubuntu-system-service unar youtube-dl zeitgeist-core
使用'sudo apt autoremove'来卸载它(它们)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 11 个软件包未被升级。
user@Phytium-Pi:~$
```

2.6.2．配置登录会话

将远程桌面默认登录会话配置为 xfce4。

配置文件，需要放在登录的用户的目录下，如放置目录为/home/<user>/.xsession，这里的 user 是计划用于远程登录的用户名，不得与飞腾派当前桌面登录的用户名重合，否则将出现闪退。

```
user@Phytium-Pi:~$ pwd
/home/user
user@Phytium-Pi:~$ touch .xsession
user@Phytium-Pi:~$ echo xfce4-session > .xsession
user@Phytium-Pi:~$
user@Phytium-Pi:~$
```

2.6.3．确认连接信息

（1）确认 xrdp 服务状态

```
$ sudo systemctl status xrdp
```

```
user@Phytium-Pi:~$ pwd
/home/user
user@Phytium-Pi:~$ touch .xsession
user@Phytium-Pi:~$ echo xfce4-session > .xsession
user@Phytium-Pi:~$
user@Phytium-Pi:~$ sudo systemctl status xrdp
● xrdp.service - xrdp daemon
     Loaded: loaded (/lib/systemd/system/xrdp.service; enabled; vendor preset: >
     Active: active (running) since Tue 2023-08-08 17:47:32 CST; 23h ago
       Docs: man:xrdp(8)
             man:xrdp.ini(5)
    Process: 2595 ExecStartPre=/bin/sh /usr/share/xrdp/socksetup (code=exited, >
    Process: 2610 ExecStart=/usr/sbin/xrdp $XRDP_OPTIONS (code=exited, status=0>
   Main PID: 2624 (xrdp)
      Tasks: 2 (limit: 1521)
     Memory: 18.3M
     CGroup: /system.slice/xrdp.service
             ├─2624 /usr/sbin/xrdp
             └─3569 /usr/sbin/xrdp

8月 09 16:28:05 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[INFO ] IPv6 not >
8月 09 16:28:05 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[INFO ] xrdp_wm_l>
8月 09 16:28:05 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[DEBUG] xrdp_wm_l>
8月 09 16:28:05 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[DEBUG] return va>
8月 09 16:28:05 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[INFO ] xrdp_wm_l>
8月 09 16:28:05 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[DEBUG] xrdp_wm_l>
8月 09 16:28:05 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[INFO ] lib_mod_l>
8月 09 16:28:05 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[DEBUG] xrdp_wm_l>
8月 09 16:28:06 Phytium-Pi xrdp[3569]: (3569)(281473663549456)[DEBUG] xrdp_mm_c>
lines 1-23
```

（2）确认板端 IP 地址

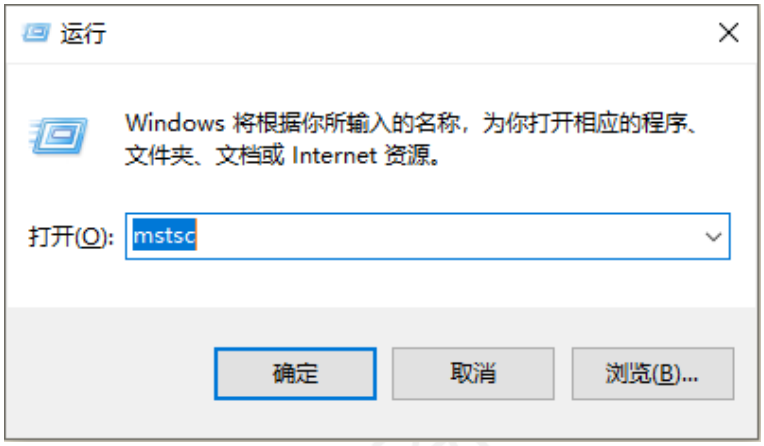
用 ifconfig 命令查看，WiFi 连接的话查看 wlan0，以太网连接查看 eth0 或者 eth1。

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.241  netmask 255.255.255.0  broadcast 192.168.100.255
        ether 14:f5:f9:87:2f:e2  txqueuelen 1000  (以太网)
        RX packets 4336  bytes 740574 (740.5 KB)
        RX errors 0  dropped 4  overruns 0  frame 0
        TX packets 6979  bytes 9111526 (9.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

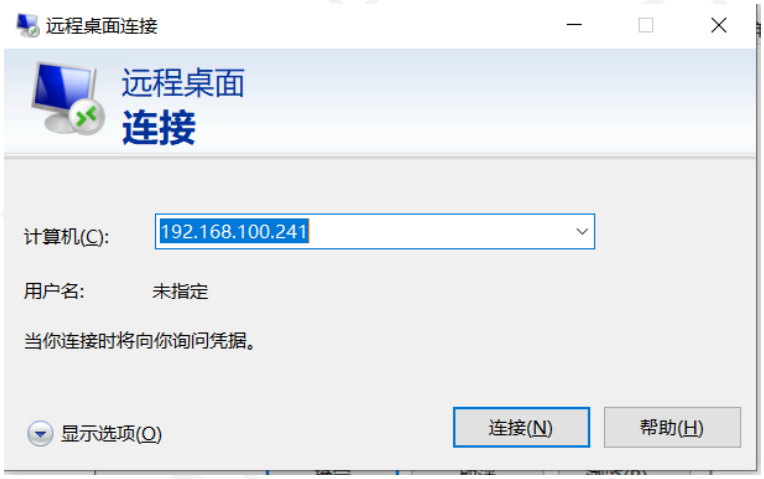
2.6.4. 远程连接

Windows 系统按照下面的步骤进行远程桌面连接：

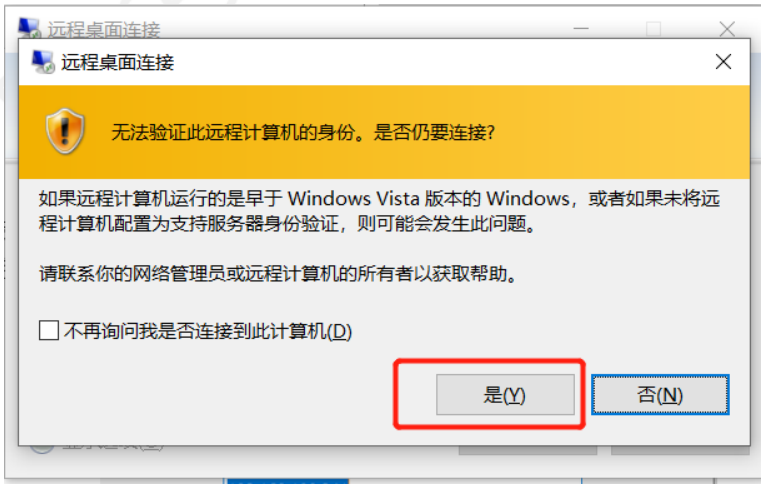
（1）使用 win+r 键打开运行解码，输入 mstsc 后回车。



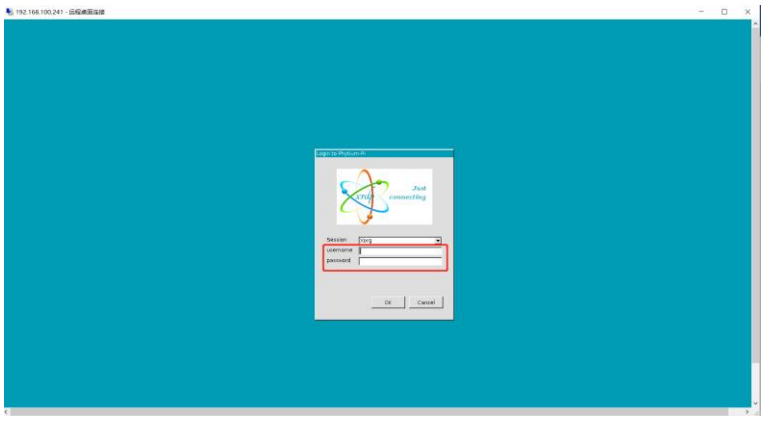
（2）输入远程桌面登录信息（IP 地址、用户名），注意这里的 IP 和用户名均为飞腾派的 IP 地址和用户名，此用户名不得与飞腾派当前桌面登录的用户名重合，否则将出现闪退。输入完成后点击连接。



（3）安全提醒确定，此时可能会弹出连接警告，需要二次进行确认，点击“是”即可。



（4）输入凭据/密码，此时可能会弹出连接用户信息确认，输入登录名和密码后点击“OK”：



（5）连接成功，输入信息正确后，一般建立连接成功，显示以下登录界面：

2.6.5．连接问题排查

下面列出了一些常见可能会导致远程桌面连接出现闪退的情况：

（1）文件配置错误

确认板端 ~/.xsession 文件内容是否正常配置，参考前面介绍的“配置登录会话”；

（2）登录用户占用

确保远程登录的用户已经在本地下线，每个登录用户只能创建一个桌面会话，如果本地使用“user”用户进行登录，那么远程桌面不能同时使用“user”用户登录桌面。串口登录使用 user 不影响，但是图形界面端登录的会影响。

（3）桌面更换重设

若对用户桌面进行更换，需要重新设置（见 2.6 章）。

（4）配置文件放置错误

配置文件，需要放在远程桌面登录的用户的目录下，如放置目录为/home/user/.xsession。

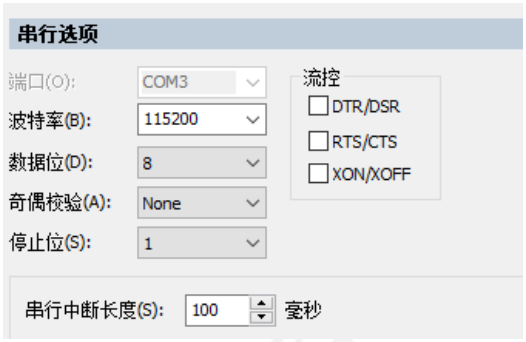
2.6.6．特别说明

请注意，由于各个远程服务器和客户端软件的功能和设置方式略有不同，上述步骤仅提供了基本的指导，具体的操作步骤请参考相关软件的文档和帮助文本。

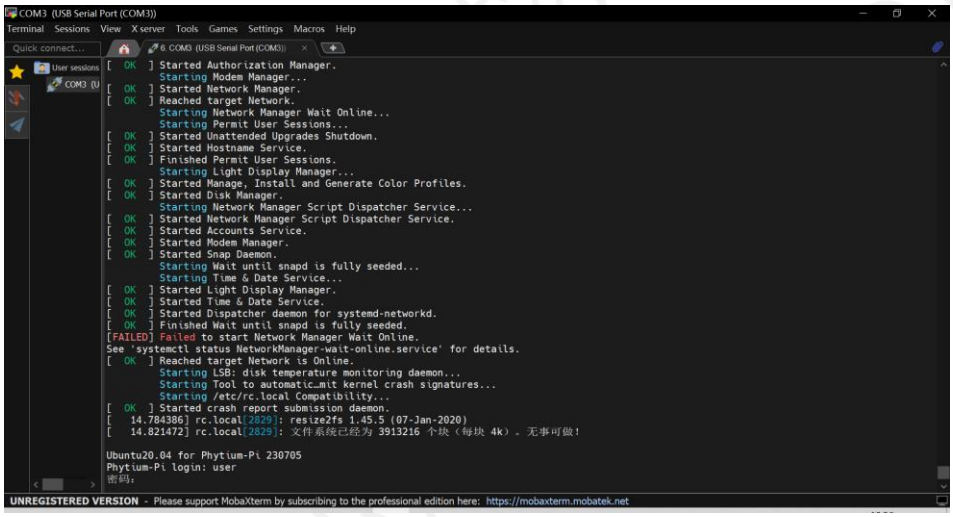
2.7．访问控制

2.7.1．串口访问

使用串口访问板卡时，调试串口位置为 J2，串口终端程序必须设置为：波特率：115200；数据位：8；奇偶校验：无；停止位：1



系统调试串口下的默认用户名：user；密码：user。Root 用户名：root；密码：root。



2.7.2．图形界面访问

使用图形界面访问，通过 HDMI 接口连接显示器后上电开机进入图形操作界面即可。

基础编程篇

3．Linux 基础

3.1．linux 常用命令

基本格式：

```
[root@localhost ~] # 命令 [选项] [参数]
```

提示：

Linux 命令，基本上遵循以上格式。

在所有的计算机文档中，在说格式的规则中，[]的意思都表示可选项。

举例 ls 命令：ls 是最常见的目录操作命令，主要作用是显示目录下的内容。

命令名称：ls。

英文原意：list。

所在路径：/bin/ls。

执行权限：所有用户。

功能描述：显示目录下的内容。

```
[root@localhost ~] # ls [选项] [文件名或目录名]
选项：
-a：显示所有文件
--color=when：支持颜色输出，when的值默认是always（总显示颜色），也可以是never（从不显示颜色）和auto（自动）
-d：显示目录信息，而不是目录下的文件
-h：人性化显示，按照我们习惯的单位显示文件大小
-i：显示文件的i节点号。理论上i节点号不重叠。
-l：长格式显示（long list）
以上是ls命令，比较常用的选项。
```

3.1.1．新手必备的 Linux 命令

（1）pwd 命令

使用 pwd 命令找出您所在的当前工作目录（文件夹）的路径。该命令将返回一个绝对（完整）路径，该路径基本上是所有以 / 开头的目录的路径。绝对路径的一个示例是 /home/username。

（2）cd 命令

要浏览 Linux 文件和目录，请使用 cd 命令。根据您所在的当前工作目录，它需要目录的完整路径或名称。假设您位于 /home/username / Documents 中，并且想要转到 Documents 的子目录 Photos。为此，只需键入以下命令：cd Photos。另一种情况是，如果您想切换到一个全新的目录，例如 /home/username / Movies。在这种情况下，您必须输入 cd，然后输入目录的绝对路径：cd /home/username / Movies。有一些快捷方式可帮助您快速导航：cd ..（带有两个点）将一个目录向上移动 cd 直接转到主文件夹 cd-（带连字符）移

动到上一个目录附带说明一下，Linux 的 shell 是区分大小写的。因此，您必须准确输入名称的目录。

（3）ls 命令

ls 命令用于查看目录的内容。默认情况下，此命令将显示当前工作目录的内容。如果要查看其他目录的内容，请键入 ls，然后键入目录的路径。例如，输入 lS / 家 / 用户名 / 文档查看的内容的文件。您可以使用 ls 命令使用以下变体：ls -R 还将列出子目录中的所有文件 ls -a 将显示隐藏的文件 ls -al 将列出文件和目录以及详细信息，例如权限，大小，所有者等。

（4）cat 命令

cat（连接的缩写）是 Linux 中最常用的命令之一。它用于在标准输出（sdout）上列出文件的内容。要运行此命令，请键入 cat，然后输入文件名及其扩展名。例如：cat file.txt。以下是使用 cat 命令的其他方法：cat> filename 创建一个新文件 cat filename1 filename2> filename3 连接两个文件（1 和 2），并将它们的输出存储在新文件中（3）将文件转换为大写或小写使用，cat filename | tr a-z A-Z >output.txt

（5）cp 命令

使用 cp 命令将文件从当前目录复制到另一个目录。例如，命令 cp scenery.jpg/home /username/ Pictures 将在您的 Pictures 目录中创建一个 Scene.jpg 副本（来自当前目录）。

（6）mv 命令

mv 命令的主要用途是移动文件，尽管它也可以用于重命名文件。mv 中的参数类似于 cp 命令。您需要输入 mv，文件名和目标目录。例如：mv file.txt/home /username/ Documents。

（7）mkdir 命令

使用 mkdir 命令创建一个新目录 - 如果键入 mkdir Music，它将创建一个名为 Music 的目录。还有一些额外的 mkdir 命令：要在另一个目录中生成新目录，请使用此 Linux 基本命令 mkdir Music / Newfile 使用 p（父级）选项在两个现有目录之间创建一个目录。例如，

mkdir -p Music / 2022 / Newfile 将创建新的 “2022” 文件。

（8）rmdir 命令

如果需要删除目录，请使用 rmdir 命令。但是，rmdir 仅允许您删除空目录。

（9）rm 命令

该 rm 命令用于删除目录以及其中的内容。如果只想删除目录（作为 rmdir 的替代方法），请使用 rm -r。注意：使用此命令时要格外小心，并仔细检查您所在的目录。这将删除所有内容，并且没有撤消操作。

（10）touch 命令

该触摸命令允许您创建通过 Linux 命令行新的空白文件。例如，输入 touch /home/username/Documents/Web.html 在 Documents 目录下创建一个名为 Web 的 HTML 文件。

（11）locate 命令

您可以使用此命令来定位文件，就像 Windows 中的搜索命令一样。此外，将 - i 参数与该命令一起使用将使其不区分大小写，因此即使您不记得其确切名称，也可以搜索文件。要搜索包含两个或多个单词的文件，请使用星号（*）。例如，locate -i school * note 命令将搜索包含单词 “school” 和 “ note” 的任何文件，无论它是大写还是小写。

（12）find 命令

在类似定位命令，使用 查找也搜索文件和目录。区别在于，您可以使用 find 命令在给定目录中查找文件。例如，find /home/-name notes.txt 命令将在主目录及其子目录中搜索名为 notes.txt 的文件。使用查找时的其他变化 是：要查找当前目录中使用的文件，请使用 find . -name notes.txt 要查找目录，请使用 /-type d -name notes. txt13. grep 命令无疑对日常使用很有帮助的另一个基本 Linux 命令是 grep。它使您可以搜索给定文件中的所有文本。为了说明这一点，grep blue notepad.txt 将在记事本文件中搜索单词 blue。包含搜索到的单词的行将被完整显示。

（13）sudo 命令

该命令是 “SuperUser Do” 的缩写，使您能够执行需要管理或超级用户权限的任务。但是，建议不要将此命令用于日常使用，因为如果您做错了一些事情，很容易发生错误。

（14）df 命令

使用 df 命令可获取有关系统磁盘空间使用情况的报告，以百分比和 KB 表示。如果要以兆字节为单位查看报告，请输入 df -m。

（15）du 命令

如果要检查文件或目录占用了多少空间，答案是 du（磁盘使用情况）命令。但是，磁盘使用情况摘要将显示磁盘块号，而不是通常的大小格式。如果要以字节，千字节和兆字节为单位查看它，请在命令行中添加 - h 参数。

（16）head 命令

所述头命令用于查看任何文本文件的第一行。默认情况下，它将显示前十行，但是您可以根据自己的喜好更改此数字。例如，如果只想显示前五行，则键入 head -n 5 filename.ext。

（17）tail 命令

该命令与 head 命令具有相似的功能，但是 tail 命令将显示文本文件的最后十行，而不是显示第一行。例如，tail -n filename.ext。

（18）diff 命令

diff 命令是差异的缩写，diff 命令逐行比较两个文件的内容。分析文件后，它将输出不匹配的行。程序员在需要进行程序更改时经常使用此命令，而不是重写整个源代码。此命令最简单的形式是 diff file1.ext file2.ext

（19）tar 命令

该 tar 命令是最常用的命令归档多个文件到一个压缩包。类似于 zip 格式常见的 Linux 文件格式，压缩是可选的。该命令具有很长的功能列表，非常复杂，例如将新文件添加到现有档案中，列出档案内容，从档案中提取内容等等。查看一些实际示例，以了解有关其他功能的更多信息。

（20）chmod 命令

chmod 是另一个 Linux 命令，用于更改文件和目录的读取，写入和执行权限。由于此命令相当复杂，因此您可以阅读完整的教程以正确执行它。

（21）chown 命令

在 Linux 中，所有文件均归特定用户所有。该 CHOWN 命令使您可以更改或文件的所有权转让给指定的用户名。例如，chown linuxuser2 file.ext 将使 linuxuser2 成为 file.ext 的所有者。

（22）jobs 命令

jobs 命令将显示所有当前作业及其状态。作业基本上是由 Shell 启动的进程。

（23）kill 命令

如果您的程序无响应，则可以使用 kill 命令手动终止它。它将向运行异常的应用发送特定信号，并指示该应用自行终止。您总共可以使用 64 个信号，但是人们通常只使用两个信号：SIGTERM（15） — 请求程序停止运行，并给它一些时间来保存其所有进度。如果在输入 kill 命令时未指定信号，则将使用此信号。SIGKILL（9） - 强制程序立即停止。未保存的进度将丢失。除了知道信号之外，您还需要知道要杀死的程序的进程标识号（PID）。如果您不知道 PID，只需运行命令 ps ux。在知道您要使用什么信号以及程序的 PID 之后，输入以下语法：kill [signal option] PID.

（24）ping 命令

使用 ping 命令检查与服务器的连接状态。例如，只需输入 ping google.com，该命令将检查您是否能够连接到 Google 并测量响应时间。

（25）wget 命令

Linux 命令行非常有用 - 您甚至可以在 wget 命令的帮助下从 Internet 下载文件。为此，只需键入 wget，然后输入下载链接即可。

（26）uname 命令

该 UNAME 命令，短期对于 Unix 名，将打印您的 Linux 系统，如计算机名称的详细信息，操作系统，内核，等等。

（27）top 命令

作为与 Windows 中的任务管理器等效的终端，top 命令将显示正在运行的进程的列表以及每个进程使用的 CPU 数量。监视系统资源使用情况非常有用，尤其是知道哪个进程由于消耗太多资源而需要终止时。

（28）history 命令

当您使用 Linux 一段时间后，您会很快注意到每天可以运行数百个命令。因此，如果您想查看之前输入的命令，运行历史记录命令特别有用。

（29）man 命令

对某些 Linux 命令的功能感到困惑吗？不用担心，您可以使用 man 命令从 Linux 的外壳程序中轻松地学习如何使用它们。例如，输入 man tail 将显示 tail 命令的手动指令。

（30）echo 命令

此命令用于将一些数据移到文件中。例如，如果要将文本 “Hello，我的名字叫 John” 添加到名为 name.txt 的文件中，则可以键入 echo Hello, my name is John >> name.txt

（31）zip，unzip 命令

使用 zip 命令将文件压缩到 zip 归档文件中，然后使用 unzip 命令从 zip 归档文件中提取压缩文件。

（32）hostname 命令

如果您想知道主机 / 网络的名称，只需键入 hostname。在末尾添加 -I 将显示您的网络的 IP 地址。

（33）useradd，userdel 命令

由于 Linux 是多用户系统，因此这意味着多个人可以同时与同一个系统进行交互。useradd 用于创建新用户，而 passwd 将密码添加到该用户的帐户。要添加名为 John 的新用户，请添加 user John，然后添加其密码类型 passwd 123456789。删除用户与添加新用户非常相似。要删除用户帐户类型，请使用 userdel UserName。

3.2．VI 编辑器的使用

3.2.1．VI 的模式介绍

基本上 vi 可以分为三种状态，分别是命令行模式（command mode）、插入模式（Insert mode）和底行模式（last line mode），各模式的功能区分如下：

（1）命令行模式（command mode）

控制屏幕光标的移动，字符、字或行的删除，移动复制某区段及进入 Insert mode 或 last line mode 下。

（2）插入模式（Insert mode）

只有在 Insert mode 下，才可以做文字输入，按「ESC」键可回到命令行模式。

（3）底行模式（last line mode）插入模式（Insert mode）

将文件保存或退出 vi，也可以设置编辑环境，如寻找字符串、列出行号等。

不过一般我们在使用时把 vi 简化成两个模式，就是将底行模式（last line mode）也算入命令行模式 command mode）。

3.2.2．VI 的基本操作

（1）进入 vi（重要）

在系统提示符号输入 vi 及文件名称后，例如$ vi myfile，就进入 vi 全屏幕编辑画面； 不过有一点要特别注意，就是进入 vi 之后，是处于「命令行模式（command mode）」，您要切换到「插入模式（Insert mode）」才能够输入文字。进入 vi 后，先不要乱动，转换到「插入模式（Insert mode）」。

（2）切换至插入模式（Insert mode）编辑文件（重要）

在「命令行模式（command mode）」下按一下字母「i」就可以进入「插入模式（Insert mode）」，这时候你就可以开始输入文字了。

（3）Insert 的切换（重要）

您目前处于「插入模式（Insert mode）」，您就只能一直输入文字，如果您发现输错了字！想用光标键往回移动，将该字删除，就要先按一下「ESC」键转到「命令行模式（command mode）」再删除文字。

（4）退出 vi 及保存文件（重要）

在「命令行模式（command mode）」下，按一下「:」冒号键进入「Last line mode」，例如：

: w filename （输入 「w filename」将文章以指定的文件名 filename 保存）

: wq (输入「wq」，存盘并退出 vi) : q! (输入 q!， 不存盘强制退出 vi)

3.2.3．VI 的实操演练

（1）移动光标

按「ctrl」+「b」：屏幕往“后”移动一页。

按「ctrl」+「f」：屏幕往“前”移动一页。

按「ctrl」+「u」：屏幕往“后”移动半页。

按「ctrl」+「d」：屏幕往“前”移动半页。

按数字「0」：移到文章的开头。

按「G」：移动到文章的最后。

按「$」：移动到光标所在行的“行尾”。

按「^」：移动到光标所在行的“行首”

按「w」：光标跳到下个字的开头

按「e」：光标跳到下个字的字尾

按「b」：光标回到上个字的开头

按「#l」：光标移到该行的第#个位置，如：5l,56l。

（2）删除文字

「x」：每按一次，删除光标所在位置的“后面”一个字符。

「#x」：例如，「6x」表示删除光标所在位置的“后面”6 个字符。

「X」：大写的 X，每按一次，删除光标所在位置的“前面”一个字符。

「#X」：例如，「20X」表示删除光标所在位置的“前面”20 个字符。

「dd」：删除光标所在行。

「#dd」：从光标所在行开始删除#行

（3）复制

「yw」：将光标所在之处到字尾的字符复制到缓冲区中。

「#yw」：复制#个字到缓冲区

「yy」：复制光标所在行到缓冲区。

「#yy」：例如，「6yy」表示拷贝从光标所在的该行“往下数”6 行文字。

「p」：将缓冲区内的字符贴到光标所在位置。注意：所有与“y”有关的复制命令都必须与“p”配合才能完成复制与粘贴功能。

（4）替换

「r」：替换光标所在处的字符。

「R」：替换光标所到之处的字符，直到按下「ESC」键为止。

（5）回复上一次操作

「u」：如果您误执行一个命令，可以马上按下「u」，回到上一个操作。按多次“u”可以执行多次回复。

（6）更改

「cw」：更改光标所在处的字到字尾处

「c#w」：例如，「c3w」表示更改 3 个字

（7）跳至指定的行

「ctrl」+「g」列出光标所在行的行号。

「#G」：例如，「15G」，表示移动光标至文章的第 15 行行首。

3.3．GCC 编译器

以源码方式安装软件，需要对源码进行编译，因此要求当前系统中存在一个可用的编译器。Linux 发行版一般都安装了 GCC 编译器，但版本一般较低。因此将以安装指定版本 GCC 编译器为例，说明使用源码安装软件的方法，其他软件源码包安装方法大同小异。

3.3.1．查看当前版本

当前 GCC 版本信息可以通过 gcc --version 获知，从中看出这是 gcc 9.4.0 版本：

```
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

如果没有安装 GCC，可以前往 GCC 官网下载指定版本的 GCC 源码安装包。

3.3.2．安装 GCC 系列环境

输入以下命令安装 gcc 系列环境：

```
$ sudo apt-get install build-essential
```

3.3.3．确认新安装 gcc 版本

输入以下命令确认 gcc 版本：

```
$ gcc -v
```

```
cek@CEK:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/aarch64-linux-gnu/9/lto-wrapper
Target: aarch64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 9.4.0-1ubuntu1~20.04.1' --with-bugurl=file:///usr/sh
are/doc/gcc-9/README.Bugs --enable-languages=c,ada,c++,go,d,fortran,objc,obj-c++,gm2 --prefix=/usr --with-gcc-major
-version-only --program-suffix=-9 --program-prefix=aarch64-linux-gnu- --enable-shared --enable-linker-build-id --li
bexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-clocale
=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-obje
ct --disable-libquadmath --disable-libquadmath-support --enable-plugin --enable-default-pie --with-system-zlib --wi
th-target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --enable-fix-cortex-a53-843419 --disable-werror
 --enable-checking=release --build=aarch64-linux-gnu --host=aarch64-linux-gnu --target=aarch64-linux-gnu
Thread model: posix
gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.1)
cek@CEK:~$
```

3.4．飞腾派环境下的 C

3.4.1．C 概述

C 语言是一门面向过程的、抽象化的通用程序设计语言，广泛应用于底层开发。C 语言能以简易的方式编译、处理低级存储器。C 语言是仅产生少量的机器语言以及不需要任何运行环境支持便能运行的高效率程序设计语言。尽管 C 语言提供了许多低级处理的功能，但仍然保持着跨平台的特性，以一个标准规格写出的 C 语言程序可在包括类似嵌入式处理器以及超级计算机等作业平台的许多计算机平台上进行编译。

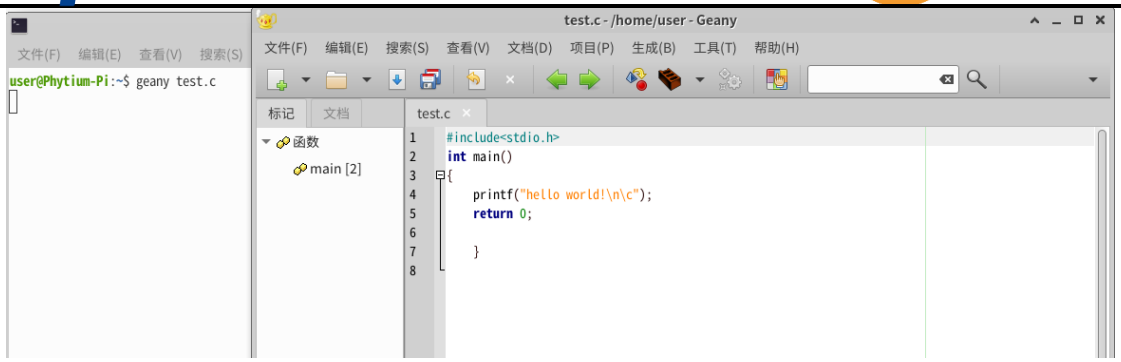
3.4.2．C 编程

（1）安装 gcc 系列环境和 geany 工具：

```
$ sudo apt-get install build-essential
$ sudo apt-get install geany
```

（2）新建源文件并编写测试代码：

```
$ touch test.c
$ geany test.c
```



（3）编译并运行

```
$ gcc test.c -o test
$ ./test
```

3.5．飞腾派环境下的 C++

3.5.1．C++概述

C++是一种计算机高级程序设计语言，由 C 语言扩展升级而产生。C++既可以进行 C 语言的过程化程序设计，又可以进行以抽象数据类型为特点的基于对象的程序设计，还可以进行以继承和多态为特点的面向对象的程序设计。

3.5.2．C++编程

（1）安装 gcc 系列环境：

```
$ sudo apt-get install build-essential
```

（2）确认 gcc 版本：

```
$ g++ -v
```

（3）新建源文件并编写测试代码：

```
$ touch test.cpp
$ geany test.cpp
```

（4）编译并运行：

```
$ g++ test.cpp -o test
$ ./test
```

3.6. 飞腾派环境下的 Python

3.6.1. Python 概述

Python 在各个编程语言中比较适合新手学习，Python 解释器易于扩展，可以使用 C 语言或 C++（或者其他可以通过 C 调用的语言）扩展新的功能和数据类型。 Python 也可用于可定制化软件中的扩展程序语言。Python 丰富的标准库，提供了适用于各个主要系统平台的源码或机器码。

3.6.2. Python 编程

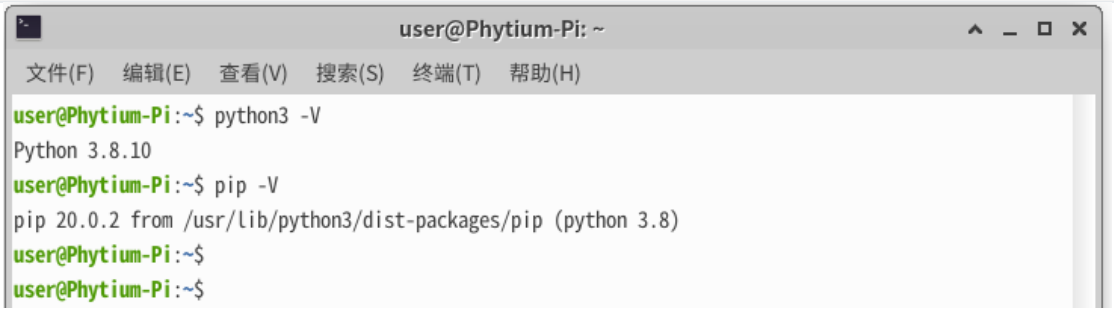
（1）安装 python 系列环境：

系统默认已经安装了 python3，可以通过下面的命令安装 pip：

```
$ sudo apt-get install python3-pip
```

（2）确认 python 与 pip 版本：

```
$ python3 -V
$ pip -V
```



（3）新建源文件并编写测试代码：

```
$ touch test.py
$ geany test.py
```

（4）编译并运行：

```
$ python3 test.py
```

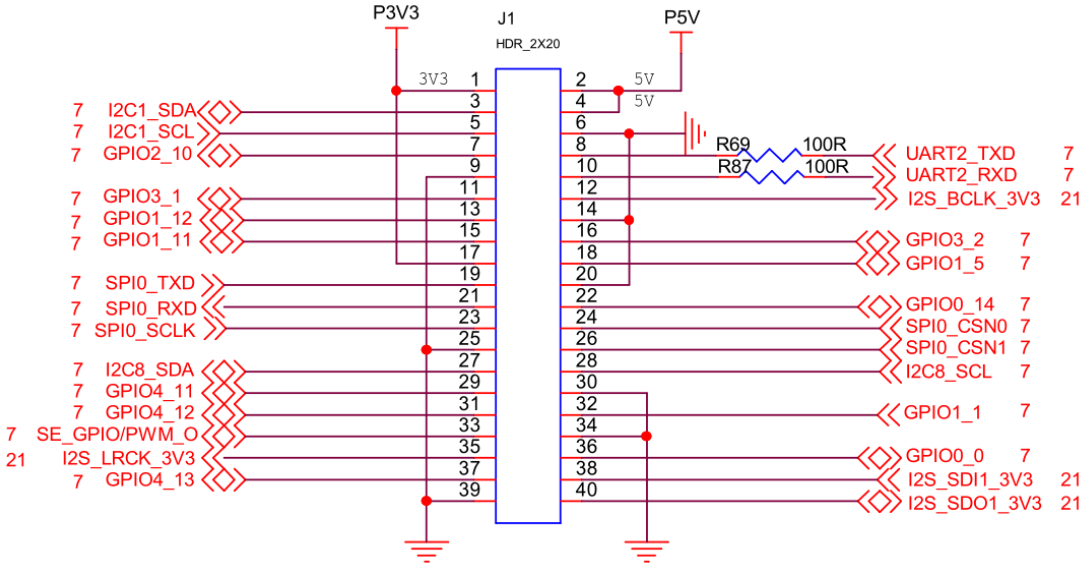
或者：

```
$ chmod +x test.py
$ ./test.py
```

4．硬件接口编程

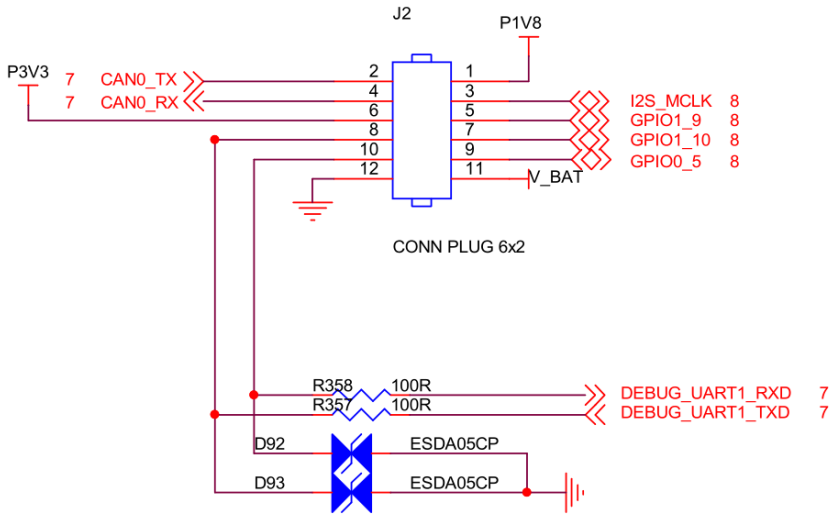
40PIN 接口原理图部分如下，有通用GPIO，I2C，UART，SPI，PWM 等接口可供使用。

其中引脚 3 和 5，引脚 8 和 10，引脚 27 和 28，引脚 11 和 16，为成对的 MIO 功能脚。MIO 是一个包含多种控制器功能的多路选择控制器，简单的理解上述配对的引脚，可以配置成 I2C 或者 UART 功能。原理图所示功能为默认配置的功能。(如需更改配置，需要指定的工具和文档，请联系商务或技术支持获取。通常情况不建议修改)



如下图，另外 12PIN 接口上的 GPIO 和 CAN 也可供使用，DEBUG_UART1 为调试串口，通过它查看系统信息和进行交互操作。其中引脚 2 和 4 位 MIO 功能脚。

1.8V IO:I2S_MCLK,GPIO0_5 ,GPIO1_9, GPIO1_10



DEBUG UART

以下操作，是对 40 pin 接口的测试记录，通过调试串口登录，基于命令行，使用 root 登录（如果用其他用户登陆，比方 user，命令前面加 sudo 以保证权限)。

如果需要从应用层操作 40pin，可参考网络上相关资料，使用通用的方法进行实现。对于 I2C/SPI/CAN，需要用户根据外挂的具体设备去实现驱动。

4.1．GPIO 接口

（1）查看节点所在目录

```
root@Phytium-Pi:~# ls /sys/class/gpio/
export          gpiochip432   gpiochip464   gpiochip496
gpiochip416   gpiochip448   gpiochip480   unexport
```

/sys/class/gpio/目录下各个文件说明：

- /sys/class/gpio/export 文件用于通知系统需要导出控制的 GPIO 引脚编号
- /sys/class/gpio/unexport 用于通知系统取消导出
- /sys/class/gpio/gpioX/direction 文件，可以写入 in （设置输入方向） 或 out （设置输出方向)
- /sys/class/gpio/gpioX/value 文件是可以读写 GPIO 状态
- /sys/class/gpio/gpiochipX 目录保存系统中 GPIO 寄存器的信息，包括每个寄存器控制引脚的起始编号，寄存器名称，引脚总数；其中 X 表示具体的引脚编号

（2）以下以操作 GPIO3_1 作为输入和 GPIO1_12 作为输出的为例，说明 GPIO 的使用方法：

①首先查看 GPIO 引脚编号

```
root@Phytium-Pi:~# cat /sys/kernel/debug/gpio
gpiochip5 : GPIOs 416-431, parent : platform/28039000 .gpio, 28039000 .gpio :
gpiochip4 : GPIOs 432-447, parent : platform/28038000 .gpio, 28038000 .gpio :
gpiochip3 : GPIOs 448-463, parent : platform/28037000 .gpio, 28037000 .gpio :
gpiochip2 : GPIOs 464-479, parent : platform/28036000 .gpio, 28036000 .gpio :
gpio-471 (                    |sel                   ) out lo
gpio-475 (                    |det                   ) in  hi IRQ
gpiochip1 : GPIOs 480-495, parent : platform/28035000 .gpio, 28035000 .gpio :
gpio-485 (                    |sysled                ) out lo
gpiochip0 : GPIOs 496-511, parent : platform/28034000 .gpio, 28034000 .gpio :
```

GPIO1_x 归属于 gpiochip1，以此类推。

GPIO1_12 归属于 gpiochip1，gpiochip1 对应的编号从 480 到 495，GPIO1_12 偏移 12，所以对应的编号为 480+12=492。同理，GPIO3_1 的编号为 448+1=449。

②导出操作节点

```
root@Phytium-Pi:~# cd /sys/class/gpio
root@Phytium-Pi:/sys/class/gpio# ls
export          gpiochip432  gpiochip464  gpiochip496
gpiochip416  gpiochip448  gpiochip480  unexport
root@Phytium-Pi:/sys/class/gpio# echo 492 > export
root@Phytium-Pi:/sys/class/gpio# echo 449 > export
root@Phytium-Pi:/sys/class/gpio# ls
export     gpio492         gpiochip432  gpiochip464  gpiochip496
gpio449  gpiochip416  gpiochip448  gpiochip480  unexport
root@Phytium-Pi:/sys/class/gpio#
```

操作过后，多出了 gpio492 和 gpio449 两个文件夹，分别对应 GPIO1_12 和 GPIO3_1。

③设置输入/输出模式 ，配置输出值/读取输入值

进入 gpio492 文件夹（GPIO1_12），设置模式为输出，输出高电平：

```
root@Phytium-Pi:/sys/class/gpio# cd gpio492
root@Phytium-Pi:/sys/class/gpio/gpio492# ls
active_low  device  direction  edge  power  subsystem  uevent  value
root@Phytium-Pi:/sys/class/gpio/gpio492# echo out > direction
root@Phytium-Pi:/sys/class/gpio/gpio492# echo 1 > value
```

此时可测量到 GPIO1_12 输出高电平。

配置为输入：

```
root@Phytium-Pi:/sys/class/gpio/gpio492# echo 0 >value
```

此时可测量到 GPIO1_12 输出低电平。

进入 gpio449 文件夹（GPIO3_1），设置模式为输入：

```
root@Phytium-Pi:/sys/class/gpio/gpio492# cd ../gpio449
root@Phytium-Pi:/sys/class/gpio/gpio449# echo in > direction
root@Phytium-Pi:/sys/class/gpio/gpio449#
```

为了测量方便，可以把 GPIO1_12 和 GPIO3_1 短接

```
root@Phytium-Pi:/sys/class/gpio/gpio449# cd ../
root@Phytium-Pi:/sys/class/gpio# echo 1 > gpio492/value
root@Phytium-Pi:/sys/class/gpio# cat gpio449/value
1
root@Phytium-Pi:/sys/class/gpio# echo 0 > gpio492/value
root@Phytium-Pi:/sys/class/gpio# cat gpio449/value
```

```
0
```

可以看到，改变 GPIO1_12 的输出电平，GPIO3_1 的获取到对应的输入电平。

4.2．PWM 接口

以 PIN32 BUZZER 脚为例，看原理图对应的是 PWM2，进行如下操作，通过配置节点参数来控制 PWM。

（1）导出节点

```
root@Phytium-Pi:~# cd /sys/class/pwm/pwmchip2/
root@Phytium-Pi:/sys/class/pwm/pwmchip2# echo 0 > export
root@Phytium-Pi:/sys/class/pwm/pwmchip2# cd pwm2/
```

（2）配置参数

设置周期值，单位为 ns，比如 20KHz 频率的周期就是 50000ns：

```
root@Phytium-Pi:/sys/class/pwm/pwmchip2/pwm2# echo 1000000 > period
```

设置的一个周期的 ON 时间，单位为 ns，也就是高电平时间：

```
root@Phytium-Pi:/sys/class/pwm/pwmchip2/pwm2# echo 500000 > duty_cycle
```

设置极性：以低电平为开启状态，还是以高电平为开启状态，一般为 normal，也就是高电平为开启状态(inversed、 normal)：

```
root@Phytium-Pi:/sys/class/pwm/pwmchip2/pwm2# echo normal > polarity
```

开启输出：

```
root@Phytium-Pi:/sys/class/pwm/pwmchip2/pwm2# echo 1 > enable
```

4.3．UART 接口

（1）查看

40PIN 上的 UART2 在系统里对应的节点为/dev/ttyAMA2，下面使用 stty 进行串口配置。

先查看 ttyAMA2 默认配置，可以看到，默认波特率 9600，如果需要查看更多信息，可以在指令后面加上-a

```
root@Phytium-Pi:~#    stty -F /dev/ttyAMA2
speed 9600 baud; line = 0;
-brkint -imaxbel
```

接下配置波特率 115200，8 位数据位，1 位停止位，无校验，关闭回显。再查看，可以看到配置修改正确：

```
root@Phytium-Pi:~# stty -F /dev/ttyAMA2 speed 115200 cs8 -cstopb -parenb    -echo
9600
root@Phytium-Pi:~# stty -F /dev/ttyAMA2
speed 115200 baud; line = 0;
-brkint -imaxbel
-echo
```

stty 的一些常用参数如下：

```
校验位
    parenb：使终端进行奇偶校验，   -parenb 则是禁止校验；
    -parodd 偶数校验
    parodd 奇数校验
数据位
    cs5、cs6、cs7 和 cs8 分别将字符大小设为 5、6、7 和 8 比特；
波特率
    speed 波特率:设置波特率
停止位
  cstopb 和 -cstopb 分别设置两个或一个停止位；
```

（2）测试

可以通过串口调试板接上 TX、RX 进行测试。这里我们为了方便，可以把 TX 和 RX 短接，串口自发自收进行测试。

用 cat 命令开启 ttyAMA2 的接收，加 & 让命令后台运行，这样我们可以继续输入命令：

```
root@Phytium-Pi:~# cat /dev/ttyAMA2 &
[1] 3512
```

用 echo 命令通过 ttyAMA2 发送数据，可以看到，串口接收到自己发送的数据：

```
root@Phytium-Pi:~# echo "uarttest"   >   /dev/ttyAMA2
root@Phytium-Pi:~# uarttest
```

断开 TX 和 RX 的连接，再测试，发出的数据就没有收到：

```
root@Phytium-Pi:~# echo "uarttest"   >   /dev/ttyAMA2
root@Phytium-Pi:~#
```

外接串口板测试的方法跟以上一样。

4.4. I2C 接口

外挂设备测试，可以使用 linux 下的通用 I2C 工具进行测试。

先安装工具：

```
sudo apt-get install i2c-tools
```

通过串口输入命令进行 I2C 读写操作，简单介绍如下：

```
i2cdetect   -l   测试所有总线
i2cdetect   -y -r 2   测试总线 2 上的设备
i2cdump -f -y 2 0x1a  读取获取总线 2 下 0x1a 的所有值

8 位地址读写例子
i2cget -f -y 2 0x1a 0x00  读取总线 2 上的地址为 0x1a 的设备的 0x00 寄存器的值
i2cset -f -y 2 0x1a 0x00 0x2   设置总线 2 上的地址为 0x1a 的设备的 0x00 寄存器的值为 0x2

16 位地址读写例子
i2ctransfer -y -f [i2c id] w2@[i2c addr] [reg_addr_h] [reg_addr_l] r1
i2ctransfer -y -f [i2c id] w3@[i2c addr] [reg_addr_h] [reg_addr_l] [val]
i2ctransfer -f -y 4 w2@0x60 0x30 0x14 r1  读取总线 4 上的地址为 0x60 的设备的 0x3014 寄存器的值
i2ctransfer -f -y 4 w3@0x60 0x30 0x14 0x00  设置总线 4 上的地址为 0x60 的设备的  0x3014 寄存器的值为 0x00
```

如果设备需要编写驱动，则用户需按实际外设编写驱动，并添加到内核或以 ko 文件进行加载，还需修改设备树 dts 文件。

假设 I2C1 外挂了型号为 ds1339 的 rtc 芯片，驱动编写可参考./drivers/rtc/rtc-ds1307.c。

设备树修改如下，添加对应的 rtc 节点，compatible 参数需要和驱动里的信息对应，i2c 地址 0x68 也要填写正确。

```
  mio1 : i2c@28016000 {
                compatible = "phytium,i2c";
                reg = <0x0 0x28016000 0x0 0x1000>;
                interrupts = <GIC_SPI 93 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&sysclk_50mhz>;
                #address-cells = <1>;
                #size-cells = <0>;
                status = "okay";


                rtc@68 {
            compatible = "dallas,ds1339";
            reg = <0x68>;
     };
};
```

同理，使用 I2C8 则修改以下节点：

```
mio8 : i2c@28024000 {
        compatible = "phytium,i2c";
        reg = <0x0 0x28024000 0x0 0x1000>;
        interrupts = <GIC_SPI 100 IRQ_TYPE_LEVEL_HIGH>;
        clocks = <&sysclk_50mhz>;
        #address-cells = <1>;
        #size-cells = <0>;
        status = "okay";

        rtc@68 {
            compatible = "dallas,ds1339";
            reg = <0x68>;
        };
};
```

4.5．SPI 接口

SPI 同 I2C一样，用户需要按照实际的外设去实现驱动，设备树需要添加节点到 spi0 节点下，之后进行驱动调试：

```
&spi0 {
    global-cs = <1>;
    status = "okay";

    flash : w25q128@0 {
        compatible = "winbond,w25q128", "jedec,spi-nor";
        spi-tx-bus-width = <1>;
        spi-rx-bus-width = <1>;
        spi-max-frequency = <12000000>;
        reg = <0x00>;
        status = "disabled";
    };

    spidev0 : spidev@0 {
        compatible = "spidev";
        reg = <0>;
        spi-max-frequency = <50000000>;
        status = "disabled";
    };
  };
```

对于手头无 SPI 外设的情况，如果要测试 SPI，可参考以下操作：

打开设备树 spidev0 的 status，编译后更新设备树，就能获得一个模拟的外设：

```
&spi0 {
        global-cs = <1>;
        status = "okay";

        flash : w25q128@0 {
                compatible = "winbond,w25q128", "jedec,spi-nor";
                spi-tx-bus-width = <1>;
                spi-rx-bus-width = <1>;
                spi-max-frequency = <12000000>;
                reg = <0x00>;
                status = "disabled";
        };

        spidev0 : spidev@0 {
                compatible = "spidev";
                reg = <0>;
                spi-max-frequency = <50000000>;
                status = "okay";
        };
};
```

编译内核下的 SPI 测试工具 spidev_test，拷贝到平台上：

```
zero@pc:~/work/phytium-linux-kernel-master-Uboot-V1 .0 .0/tools/spi$
ls
Build       Makefile       spidev_fdx .c        spidev_fdx .o    spidev_test .c
spidev_test .o
include    spidev_fdx    spidev_fdx-in .o    spidev_test      spidev_test-in .o
zero@pc:~/work/phytium-linux-kernel-master-Uboot-V1 .0 .0/tools/spi$
file spidev_test
spidev_test : ELF 64-bit LSB shared object, ARM aarch64, version 1
(SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64 .so .1, for
GNU/Linux 3 .7 .0, BuildID[sha1]=e73ad2acb6cc927b1a6d26eaf39401dc445d3d1a,
   with debug_info, not stripped
```

短接 SPI 的 TX 和 RX，运行测试工具：

```
root@Phytium-Pi:~# spidev_test -D /dev/spidev0 .0 -v
spi mode : 0x0
bits per word : 8
max speed : 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF F0 0D     |  ......@.....................
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF F0 0D     |  ......@.....................
   root@Phytium-Pi:~#
```

可以看到，短接 TX RX 后，RX 收到的数据和 TX一样，说明 SPI 自发自收正常；通过示波器也能看到 SPI 数据传送波形。

断开 SPI 的 TX RX ，再运行测试，RX 收不到 TX 的数据：

```
root@Phytium-Pi:~# spidev_test -D /dev/spidev0 .0 -v
spi mode : 0x0
bits per word : 8
max speed : 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF F0 0D     |  ..... @.▒.....▒
RX | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
  FF  FF FF FF FF FF FF FF FF FF     | .....
```

4.6．CAN 接口

板上的 CAN 接口是芯片里 CAN 控制器直接输出，如果要挂 CAN 设备，还需要外接 CAN PHY 芯片进行电压转换。以下为模拟测试：

系统默认开了 can0 节点：

```
root@Phytium-Pi:~# ifconfig -a
can0 : flags=128<NOARP>    mtu 72
          unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
txqueuelen 10    (
    UNSPEC)
          RX packets 0    bytes 0 (0 .0 B)
          RX errors 0    dropped 0    overruns 0    frame 0
          TX packets 0    bytes 0 (0 .0 B)
          TX errors 0    dropped 0 overruns 0    carrier 0    collisions 0
   device interrupt 81
```

安装 CAN 测试工具：

```
root@Phytium-Pi:~# sudo apt-get install can-utils
```

配置缓冲区大小：

```
root@Phytium-Pi:~# echo    4096 > /sys/class/net/can0/tx_queue_len
```

配置 CAN 波特率：

```
root@Phytium-Pi:~# ip link set can0 up type can bitrate 500000 sample-
point 0 .75 dbitrate 4000000 dsample-point 0 .8 fd on
```

CAN 发送，工具随机产生了数据并发送，用示波器测量能看到波形：

```
root@Phytium-Pi:~# cangen can0 -v -b -g 20 &    发送
can0    71F##1 .07 .6F .BB .30 .2D .C7 .39 .1D
can0    5DE##1
can0    3FB##1 .CC .8F .D3 .0D .A1 .A4 .79 .5F .CC .8F .D3 .0D .A1 .A4 .79 .5F
can0    088##1 .36 .84 .61 .56 .5B .9F .12 .06
```

CAN 接收：

```
root@Phytium-Pi:~# candump can0 &
```

4.7．USB 接口

（1）使用 lsusb 命令可查看当前挂载的 USB 设备，可以看到，分别挂载了一个 USB 键盘，一个读卡器，一个 U 盘，还有一个 USB 鼠标：

```
user@Phytium-Pi:~$ lsusb
Bus 003 Device 002: ID 09da:2268 A4Tech Co., Ltd. USB Keyboard
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 14cd:1212 Super Top microSD card reader (SY-T18)
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 006 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 002: ID 0951:1666 Kingston Technology DataTraveler 100
G3/G4/SE9 G2
Bus 005 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 093a:2510 Pixart Imaging, Inc. Optical Mouse
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

（2）lsusb 命令参数用法如下，可根据需要在使用时添加：

```
user@Phytium-Pi:~$ lsusb -h
Usage: lsusb [options]...
List USB devices
-v, --verbose
Increase verbosity (show descriptors)
-s [[bus]:][devnum]
Show only devices with specified device and/or
bus numbers (in decimal)
-d vendor:[product]
Show only devices with the specified vendor and
product ID numbers (in hexadecimal)
-D device
Selects which device lsusb will examine
-t, --tree
Dump the physical USB device hierarchy as a tree
-V, --version
Show version of program
-h, --help
Show usage and help
```

也可使用其他 USB 设备进行测试。

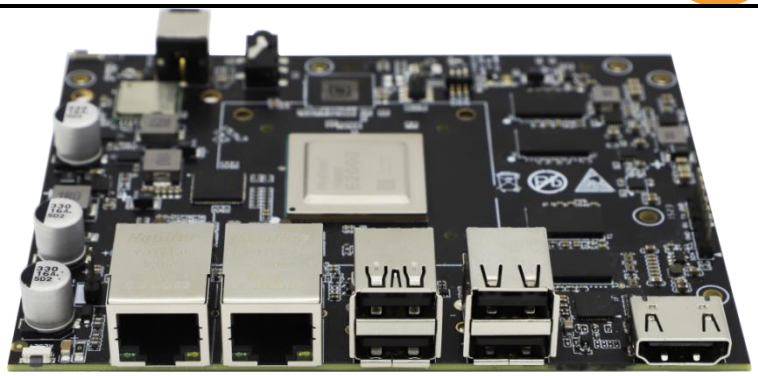
4.8．以太网接口

（1）查看网卡信息

使用 ifconfig 查看所有网卡信息：

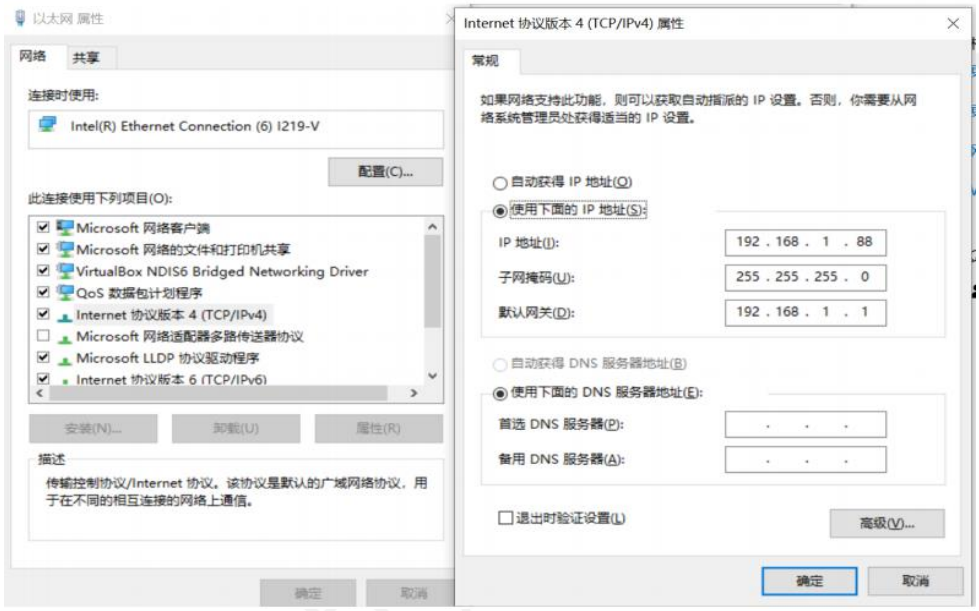
```
user@Phytium-Pi:~$ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 00:11:22:33:44:55 txqueuelen 1000 (以太网)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)Show usage and help
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 86 base 0xc000

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
ether 10:22:33:44:55:66 txqueuelen 1000 (以太网)
RX packets 6529 bytes 8668308 (8.6 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 2529 bytes 185405 (185.4 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 94 base 0xe000
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
loop txqueuelen 1000 (本地环回)
RX packets 1203 bytes 88731 (88.7 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1203 bytes 88731 (88.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 10:bb:f3:1c:2d:04 txqueuelen 1000 (以太网)
RX packets 0 bytes 105954 (105.9 KB)
RX errors 0 dropped 297 overruns 0 frame 0
TX packets 0 bytes 7847 (7.8 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
wlan1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 12:bb:f3:1c:2d:04 txqueuelen 1000 (以太网)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

ethx 为以太网卡（左边为 eth1，右边为 eth0），wlanx 为 wifi 网卡。



（2）连接以太网

使用网线连接以太网，另一端如果是连接到网络能自动获取 IP。这里我们的另一端连接的是 PC 电脑的网卡，把 PC 网卡指定 IP 为 192.168.1.88：



使用 ifconfig 命令把开发板的以太网 IP 指定为同一网段的 192.168.1.66：

```
user@Phytium-Pi:~$ sudo ifconfig eth0 192.168.1.66
```

查看是否配置成功：

```
user@Phytium-Pi:~$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.1.66 netmask 255.255.255.0 broadcast 192.168.1.255
ether 00:11:22:33:44:55 txqueuelen 1000 (以太网)
RX packets 84 bytes 7827 (7.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 58 bytes 11522 (11.5 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 86 base 0xc000
```

（3）使用 ping 命令测试通讯

ping PC 端的地址，可见已经能够正常通讯：

```
user@Phytium-Pi:~$ ping 192.168.1.88
PING 192.168.1.88 (192.168.1.88) 56(84) bytes of data.
64 字节，来自 192.168.1.88: icmp_seq=1 ttl=128 时间=0.340 毫秒
64 字节，来自 192.168.1.88: icmp_seq=2 ttl=128 时间=0.889 毫秒
64 字节，来自 192.168.1.88: icmp_seq=3 ttl=128 时间=0.468 毫秒
64 字节，来自 192.168.1.88: icmp_seq=4 ttl=128 时间=0.573 毫秒
64 字节，来自 192.168.1.88: icmp_seq=5 ttl=128 时间=0.473 毫秒
64 字节，来自 192.168.1.88: icmp_seq=6 ttl=128 时间=0.442 毫秒
```

之后就可以通过网线，在 PC 端对开发板进行各种操作。

4.9．WiFi

WiFi 连接可使用系统桌面进行配置，跟普通 PC 一样，以下是对命令行配置 WiFi 的演示：

（1）查看 WiFi 信息

使用 nmcli 命令 扫描 WiFi：

```
sudo nmcli dev wifi
```

结果如下，可以看到 WiFi 的 SSID 和速率信号强度等信息：

```
user@Phytium-Pi:~$ sudo nmcli dev wifi
IN-USE BSSID SSID MODE CHAN RATE SIGNAL
BA
RS SECURITY
D6:0D:C5:70:EA:B8 zero_test Infra
13 130 Mbit/s 87 ▂▄▆█ WPA2
94:3B:B0:36:86:90 CECport_Guest Infra
149 270 Mbit/s 85 ▂▄▆█ WPA2
94:3B:B0:36:86:81 CECport_Office Infra
36 540 Mbit/s 84 ▂▄▆█ WPA2 802.1X
94:3B:B0:36:86:80 CECport_Guest Infra
36 540 Mbit/s 82 ▂▄▆█ WPA2
94:3B:B0:36:86:91 CECport_Office Infra
149 270 Mbit/s 82 ▂▄▆█ WPA2 802.1X
94:3B:B0:36:86:A1 CECport_Office Infra
6 130 Mbit/s 70 ▂▄▆_ WPA2 802.1X
94:3B:B0:36:86:A0 CECport_Guest Infra
6 130 Mbit/s 70 ▂▄▆_ WPA2
C8:14:B4:B8:04:18 ChinaNet-chx6-5G Infra
```

```
36 270 Mbit/s 55 ▂▄__ WPA1 WPA2
74:69:4A:50:F0:47 CU_WseY_5G Infra
161 270 Mbit/s 55 ▂▄__ WPA1 WPA2
14:51:7E:E4:E9:CF CEC_NVIDIA_1_5G Infra
44 540 Mbit/s 52 ▂▄__ WPA1 WPA2
74:69:4A:50:F0:48 CU_WseY Infra
```

（2）建立连接

使用一下命令连接 zero_test 这个网络，可见连接成功：

```
user@Phytium-Pi:~$ sudo nmcli dev wifi connect "SSID" password "PWD" ifname
wlan0
成功用 "wlan0cf4937b3-ce4b-4a14-8eb4-bb086373e59e" 激活了设备 ""。
```

注意：SSID 和 PWD 应换成实际的 WIFI 名和 WIFI 密码。

```
sudo nmcli dev wifi connect "SSID" password "PWD" ifname wlan0
```

（3）查看结果

查看 wlan0，可以看到已经被分配 IP，通过 ping 一个外网地址，确定已经能够工作：

```
user@Phytium-Pi:~$ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.69.11 netmask 255.255.255.0 broadcast 192.168.69.255
蓝牙
蓝牙连接可使用系统桌面进行配置，跟普通 PC 一样，推荐使用桌面进行配置。以下是对命令行配置
蓝牙的演示
使用 hciconfig 查看蓝牙状态，可见已经在工作
使用 bluetoothctl 进入蓝牙命令行操作
使用 scan on 可以扫描到周围的蓝牙设备，scan off 停止扫描
ether 10:bb:f3:1c:2d:04 txqueuelen 1000 (以太网)
RX packets 42 bytes 7749 (7.7 KB)
RX errors 0 dropped 1 overruns 0 frame 0
TX packets 41 bytes 5008 (5.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
user@Phytium-Pi:~$ ping wwww.baidu.com
PING ps_other.a.shifen.com (110.242.68.66) 56(84) bytes of data.
64 字节，来自 110.242.68.66 (110.242.68.66): icmp_seq=1 ttl=51 时间=84.0 毫秒
64 字节，来自 110.242.68.66 (110.242.68.66): icmp_seq=2 ttl=51 时间=80.4 毫秒
64 字节，来自 110.242.68.66 (110.242.68.66): icmp_seq=3 ttl=51 时间=81.0 毫秒
64 字节，来自 110.242.68.66 (110.242.68.66): icmp_seq=4 ttl=51 时间=77.5 毫秒
```

4.10．蓝牙

蓝牙连接可使用系统桌面进行配置，跟普通 PC 一样，推荐使用桌面进行配置。以下是对命令行配置蓝牙的演示（也可以使用其他方法测试）。

（1）查看状态

使用 hciconfig 查看蓝牙状态，可见已经在工作：

```
user@Phytium-Pi:~$ hciconfig
hci0: Type: Primary Bus: UART
BD Address: 10:BB:F3:1D:3E:6A ACL MTU: 1021:8 SCO MTU: 255:12
UP RUNNING
RX bytes:1586 acl:0 sco:0 events:58 errors:0
TX bytes:6393 acl:0 sco:0 commands:59 errors:0
```

（2）进入蓝牙操作

使用 bluetoothctl 进入蓝牙命令行操作：

```
user@Phytium-Pi:~$ bluetoothctl
Agent registered
[bluetooth]#
```

（3）查看蓝牙配备

使用 scan on 可以扫描到周围的蓝牙设备，scan off 停止扫描：

```
[bluetooth]# scan on
Discovery started
[CHG] Controller 10:BB:F3:1D:3E:6A Discovering: yes
[NEW] Device 64:43:BE:C2:3A:3F 64-43-BE-C2-3A-3F
[NEW] Device 48:74:12:24:11:0A OnePlus 10 Pro
[CHG] Device 48:74:12:24:11:0A LegacyPairing: yes
[CHG] Device 34:7D:F6:C4:32:85 RSSI: -30
[CHG] Device 34:7D:F6:C4:32:85 TxPower: 12
[NEW] Device 8C:1D:96:B9:C9:53 cecport-DK-ZHAN620
[CHG] Device 48:74:12:24:11:0A RSSI: -62
[CHG] Device 48:74:12:24:11:0A LegacyPairing: no
[CHG] Device 48:74:12:24:11:0A RSSI: -70
[CHG] Device 48:74:12:24:11:0A LegacyPairing: yes
[CHG] Device 8C:1D:96:B9:C9:53 LegacyPairing: yes
[NEW] Device 46:76:4E:35:31:4F 46-76-4E-35-31-4F
[CHG] Device 88:52:EB:31:0B:FA RSSI: -30
[CHG] Device 88:52:EB:31:0B:FA LegacyPairing: yes
[CHG] Device 48:74:12:24:11:0A LegacyPairing: no
```

```
[CHG] Device 88:52:EB:31:0B:FA RSSI: -38
[CHG] Device 88:52:EB:31:0B:FA LegacyPairing: no
[CHG] Device 88:52:EB:31:0B:FA RSSI: -30
[CHG] Device 8C:1D:96:B9:C9:53 LegacyPairing: no
[CHG] Device 48:74:12:24:11:0A RSSI: -62
[CHG] Device 48:74:12:24:11:0A RSSI: -72
[CHG] Device 8C:1D:96:B9:C9:53 LegacyPairing: yes
[CHG] Device 48:74:12:24:11:0A LegacyPairing: yes
[CHG] Device 48:74:12:24:11:0A RSSI: -62
[bluetooth]# scan off
```

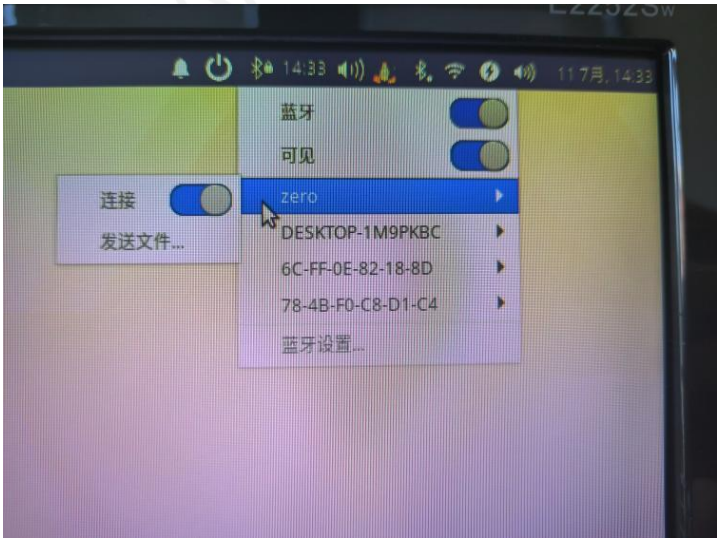
（4）指定建立连接

使用 connect 命令可指定需要连接的蓝牙 MAC 地址，这里连接的是一部手机的蓝牙 MAC 地址：

```
[bluetooth]# connect 88:52:EB:31:0B:FA
Attempting to connect to 88:52:EB:31:0B:FA
[CHG] Device 88:52:EB:31:0B:FA Connected: yes
```

（5）连接成功

连接成功后，会需要在两端（系统桌面和手机）进行配对确认，确认成功后，配对成功，如下：

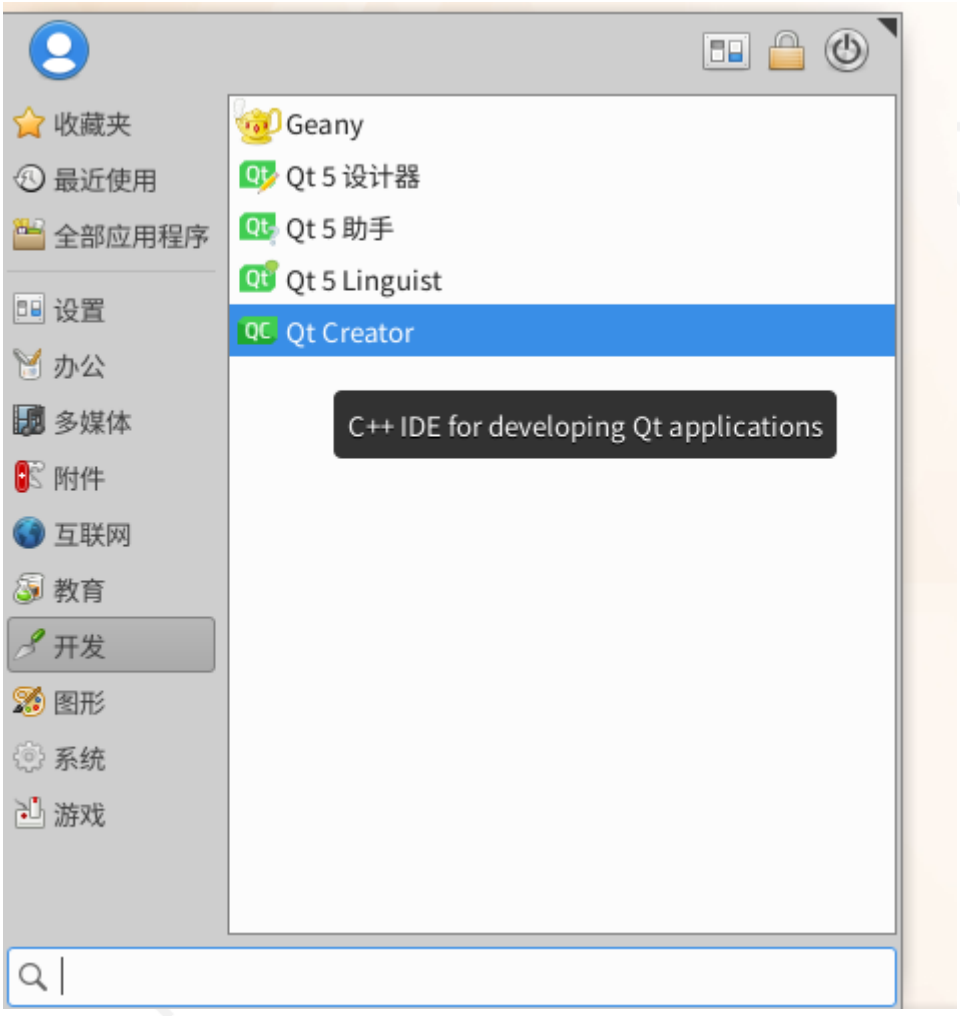


5．QT 编程

5.1．Qt Creator 简介

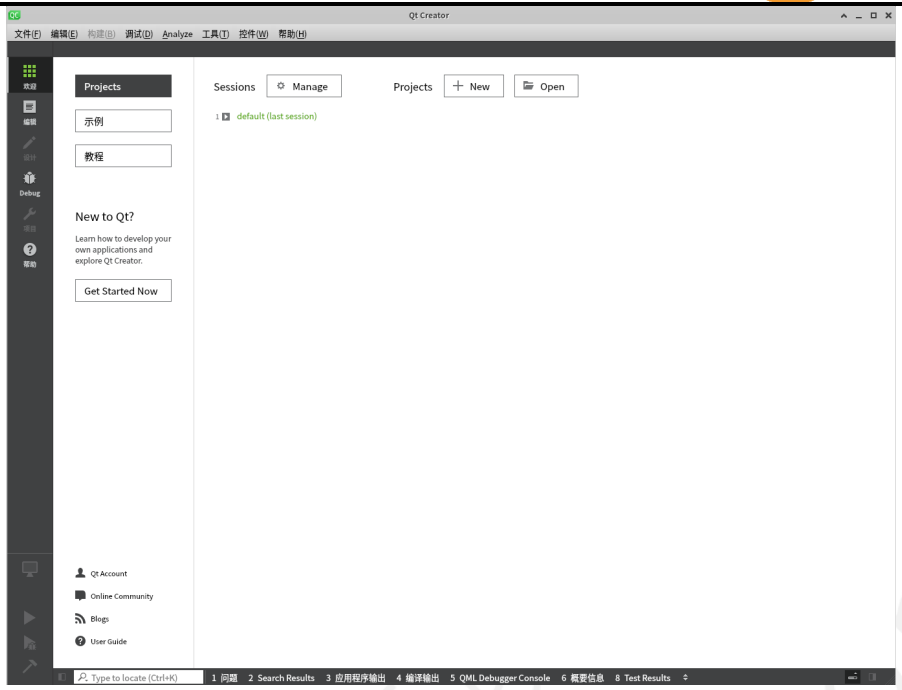
Qt 是一个由 Qt Company 开发的跨平台 C++图形用户界面应用程序开发框架。它既可以开发 GUI 程序，也可用于开发非 GUI 程序，比如控制台工具和服务器。Qt 是面向对象的框架，使用特殊的代码生成扩展以及一些宏，Qt 很容易扩展，并且允许真正地组件编程。

Qt Creator 包括项目生成向导、高级的 C++代码编辑器、浏览文件及类的工具、集成了 Qt Designer、Qt Assistant、Qt Linguist、图形化的 GDB 调试前端，集成 qmake 构建工具等。



桌面系统中已经集成了 Qt 开发环境，在开始菜单栏选择“开发”一栏即可找到 Qt Creator。

打开 Qt Creator 进行 Qt 应用开发。

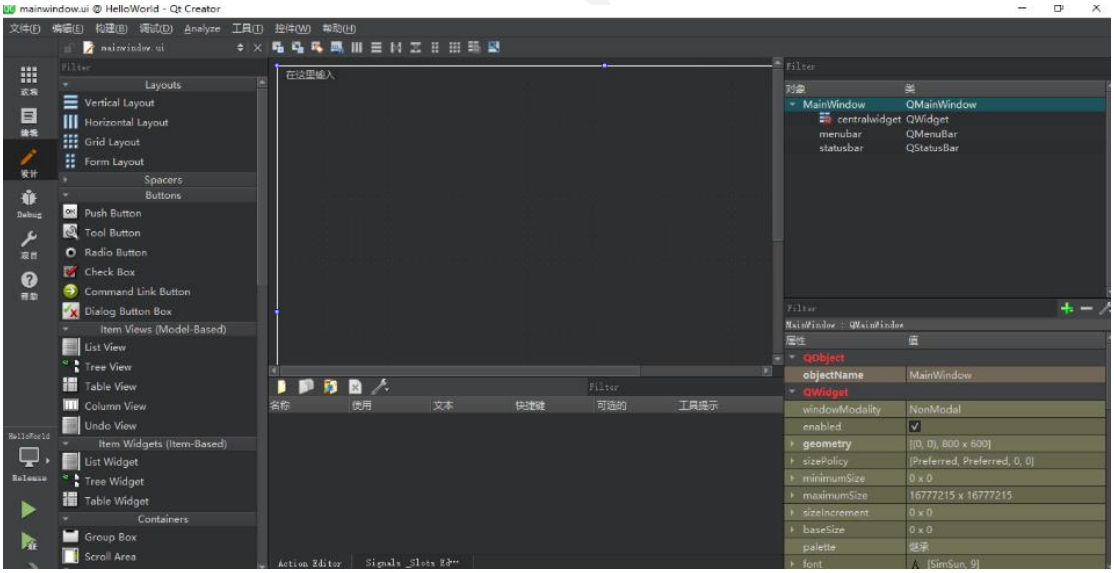


5.2．QT 界面设计

5.2.1．熟悉 Qt Designer

Qt Designer 是 Qt 专为界面设计做的软件，使得用户能够通过拖拽的方式直接布置界面，然后 Qt Creator 可以将 Qt Designer 的界面转换成 C++代码。Qt 也将 Qt Designer 内置在了 Qt Creator 中，使得用户可以在一个编辑器中来回切换界面与代码。

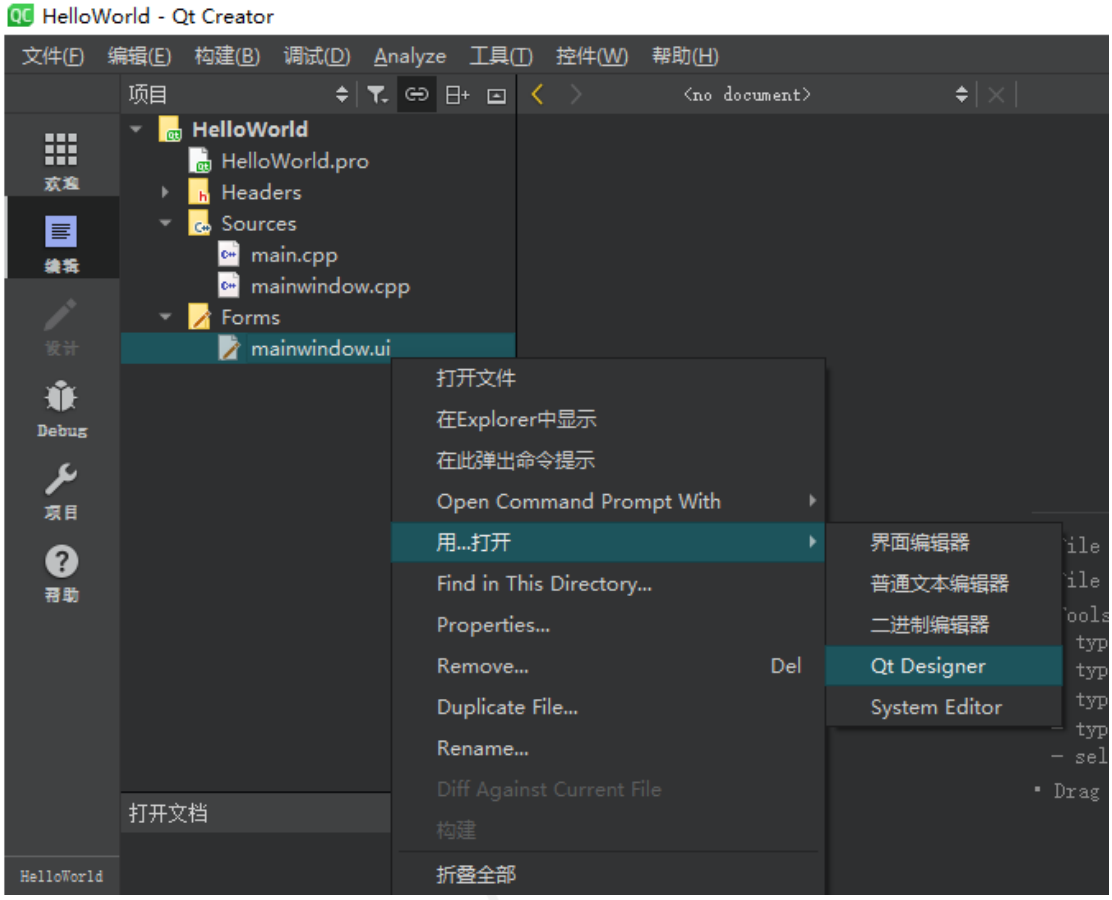
我们在之前新建的 HelloWorld 项目中，双击 mainwindow.ui，出现以下界面：



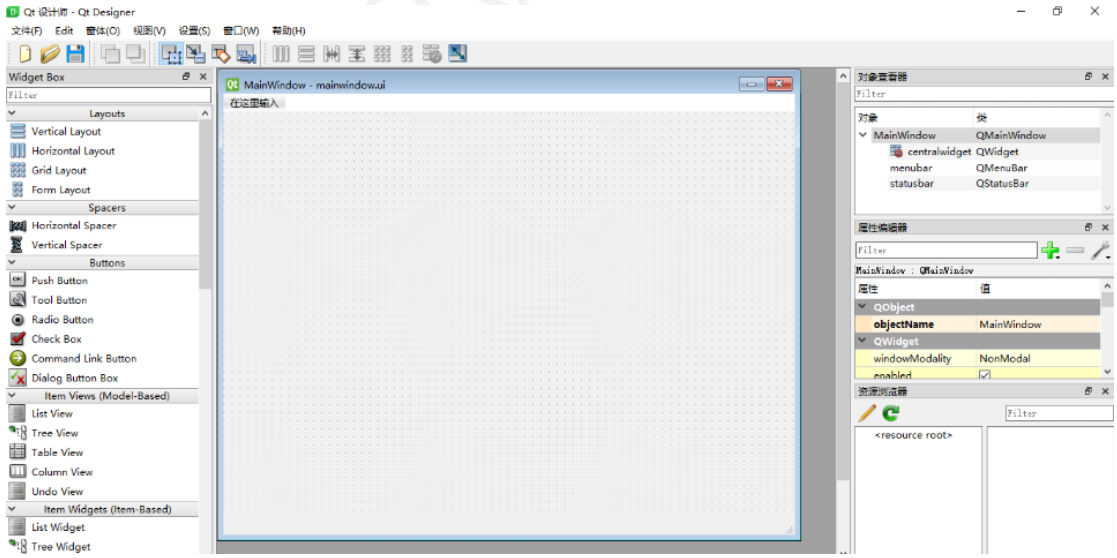
这就是嵌入在 Qt Creator 中的 Qt Designer 的样子，但是由于我们在使用 Qt Creator 编程的时候，都偏爱黑色主题，这就使得内置的 Qt Designer 黑乎乎的，控件拖拽在上面也不太清

晰，所以推荐大家单独打开 Qt Designer 来进行界面设计，单独打开的 Qt Designer 是白色主题，界面比较清晰，如果你的电脑是双屏的话，一个屏编写代码，一个屏设计界面。

如何在项目中快速地打开 Qt Designer 呢，可以这么操作：将鼠标放置在左侧项目目录中的 mainwindow.ui 上右键，选择“用...打开”，然后选择 Qt Designer，如下图所示：



打开的 Qt Designer 软件如下图所示：



即使你从没使用过这个软件，你也能一眼看懂软件各组成部分的内容。我们将界面精简一

下，去掉不常用的或暂时用不到的部分，将软件右下角的信号/槽编辑器、动作编辑器、资源浏览器子界面都叉掉，整个软件就剩下顶部的菜单工具栏，左侧的控件区，中间的界面布局区和右侧的对象查看器、属性编辑器。使用 Qt Designer 设计界面的流程大致为：

① 从控件区拖拽需要的控件到界面布局区；

② 随后在对象查看器中能够看到界面中存在的控件和控件的顺序；

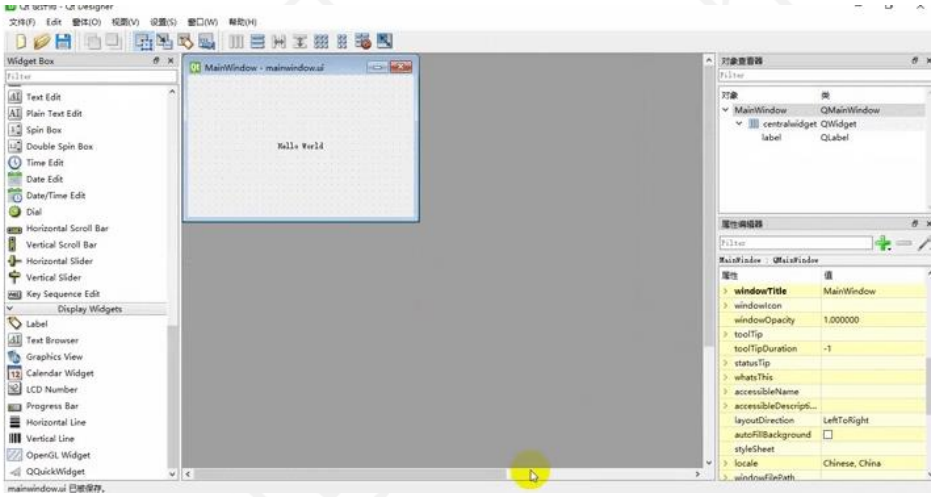
③ 然后在属性编辑器中编辑控件的属性；

④ 点击顶部的工具栏中的布局工具对界面就行布局；

⑤ 记得保存，否则 Qt Creator 无法获取最新的界面变动。

5.2.2．拖拽控件进行布局

接下来使用 Qt Designer 设计界面，在窗口中显示“Hello World”字样，然后用 Qt Creator 编译程序生成窗口。暂时不用菜单栏和状态栏，先把它们去掉，然后按照刚刚总结的五个步骤，操作如下图所示：



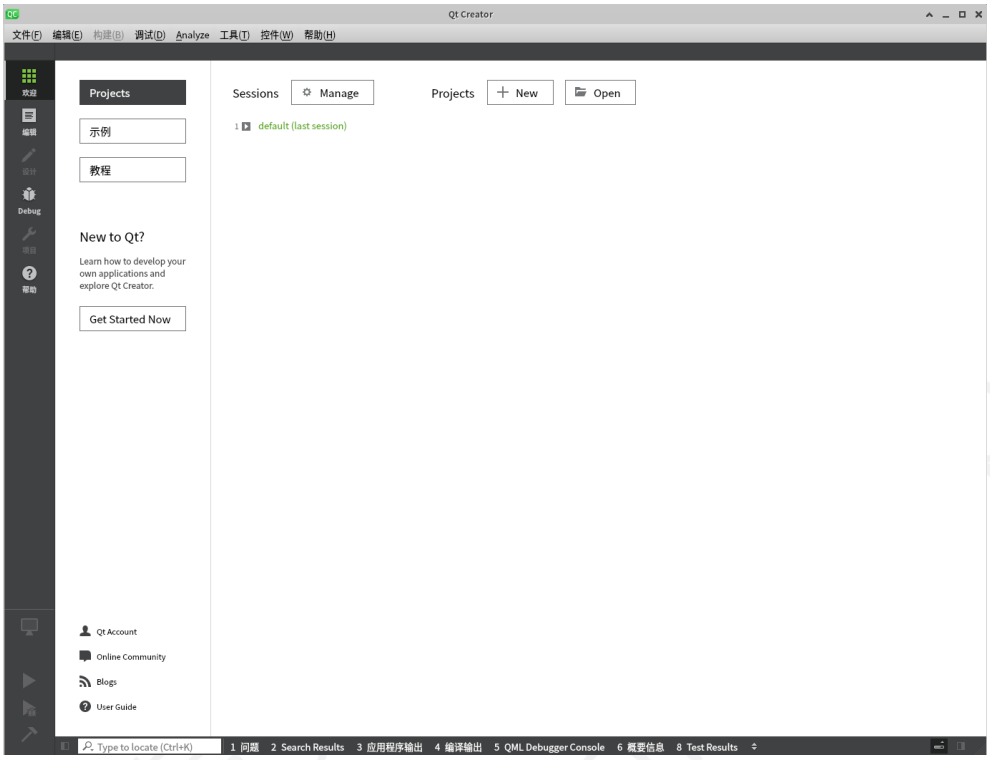
放置一个 Label 控件，设置 Label 的文本属性为“Hello World”，水平对齐为中心对齐，然后点击窗口，即选中窗口，接着点击工具栏中的“水平布局”，就将 Label 填充到了窗口中，当拉伸窗口进行窗口缩放时，“Hello World”始终在窗口中心。设计完界面之后（一定记得保存），再切换到 Qt Creator 直接运行程序，等待编译完成运行，显示如下窗口：

到此，使用 Qt Designer 进行界面设计的流程就完成了。

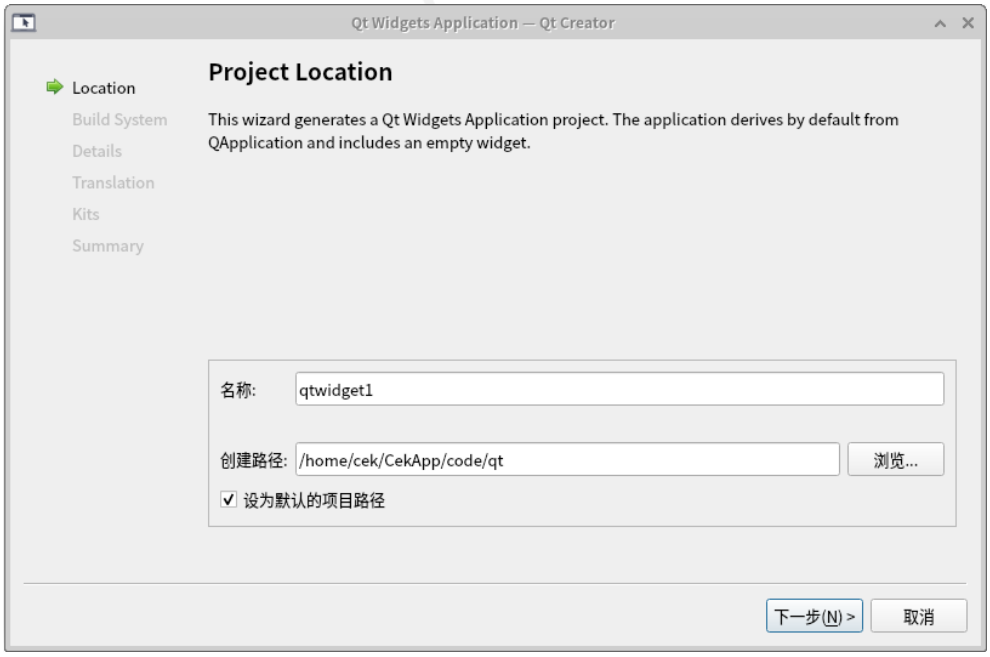
5.3．QT 应用设计

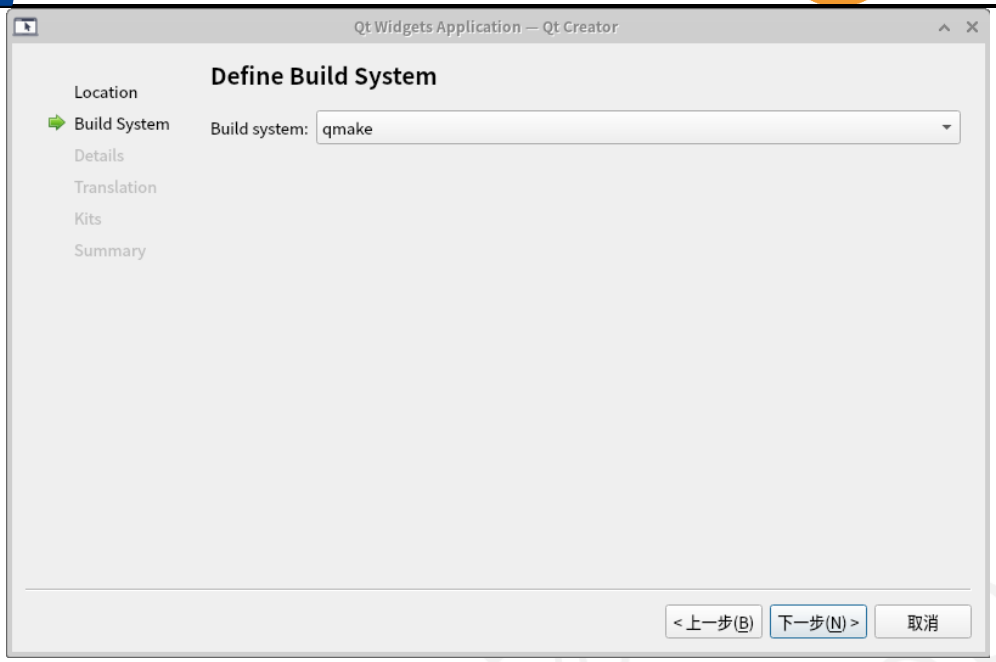
5.3.1．QtWidget 示例

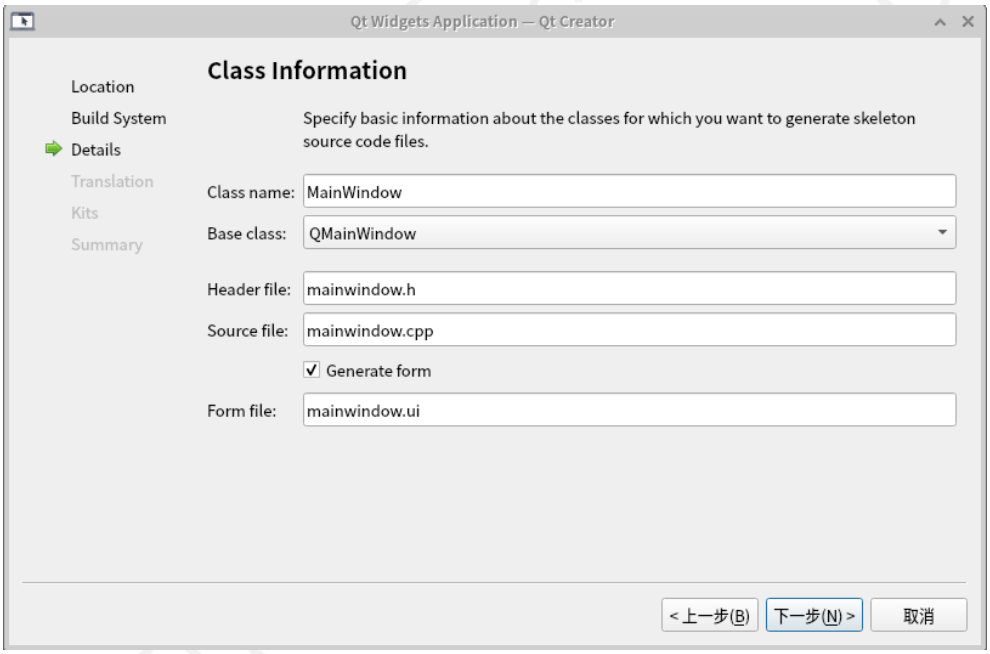
（1）在主页点击“New”新建工程：

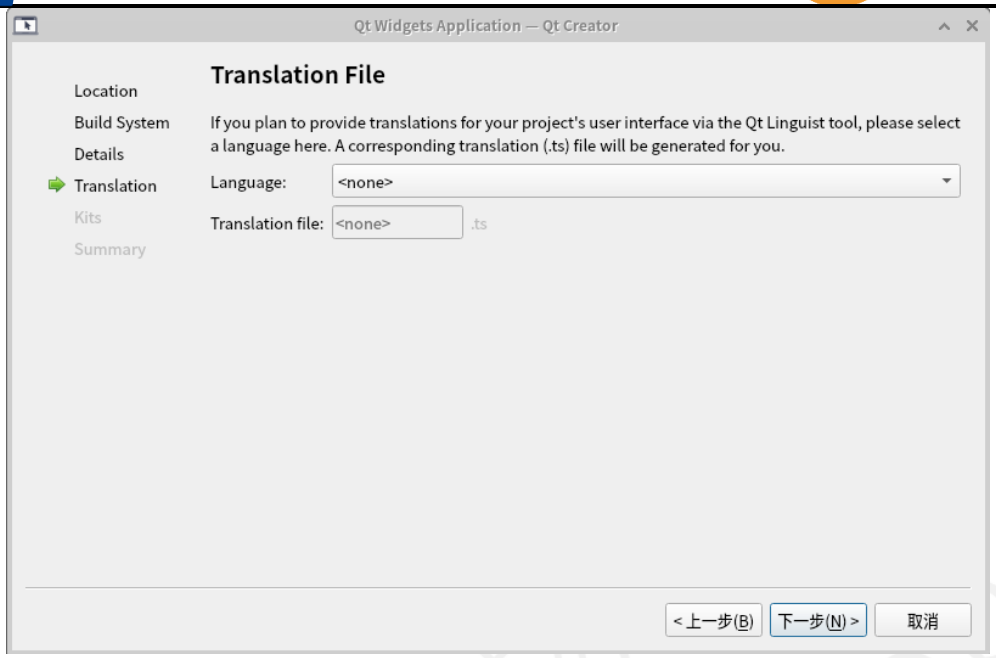


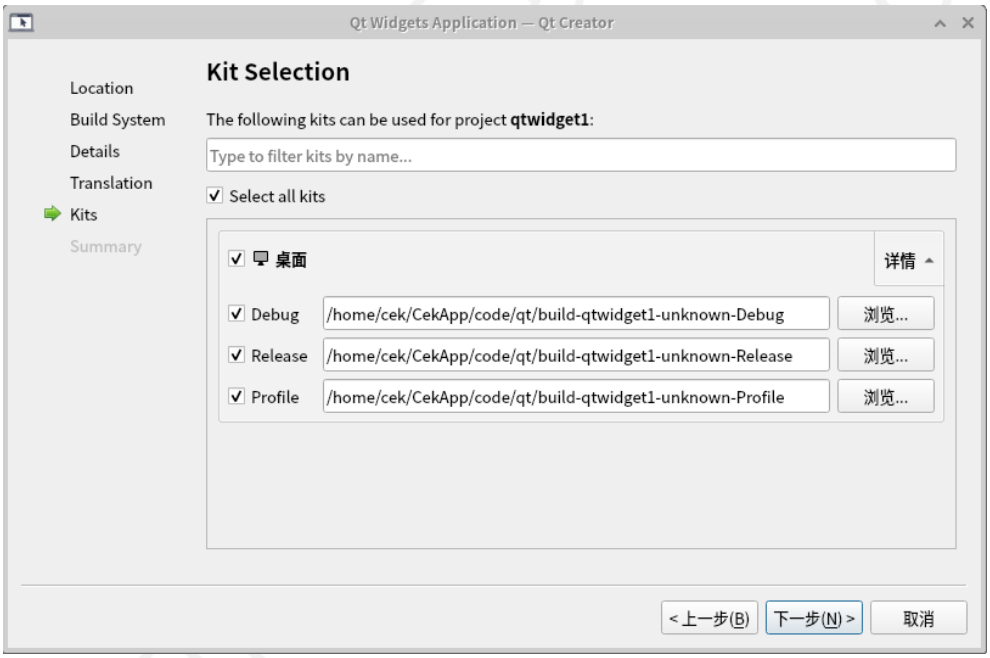
（2）选择“Qt Widgets Application”创建 QtWidget 桌面应用，按默认配置依次点击“下一步”创建：

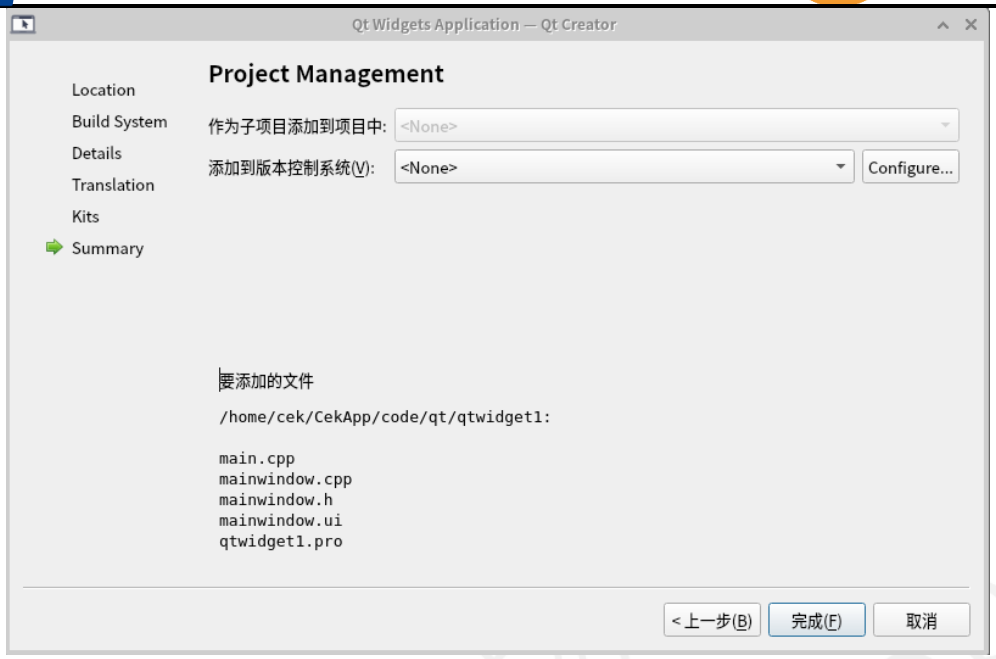



Qt Widgets Application — Qt Creator
Location
Build System
Details
Translation
Kits
Summary
Define Build System
Build system:
qmake
< 上一步(B)
下一步(N) >
取消

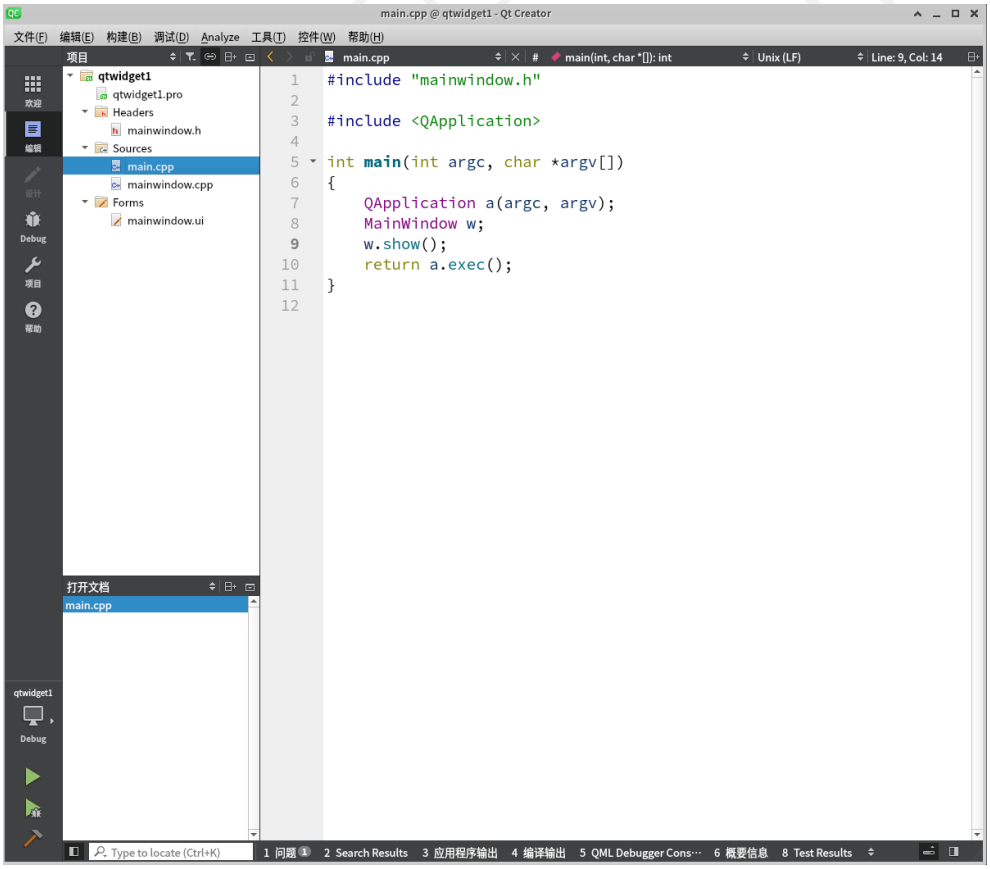

Qt Widgets Application — Qt Creator
Location
Build System
Details
Translation
Kits
Summary
Class Information
Specify basic information about the classes for which you want to generate skeleton source code files.
Class name:
MainWindow
Base class:
QMainWindow
Header file:
mainwindow.h
Source file:
mainwindow.cpp
Generate form
Form file:
mainwindow.ui
< 上一步(B)
下一步(N) >
取消


Qt Widgets Application — Qt Creator
Location
Build System
Details
Translation
Kits
Summary
Translation File
If you plan to provide translations for your project's user interface via the Qt Linguist tool, please select a language here. A corresponding translation (.ts) file will be generated for you.
Language:
<none>
Translation file:
<none>
.ts
< 上一步(B)
下一步(N) >
取消

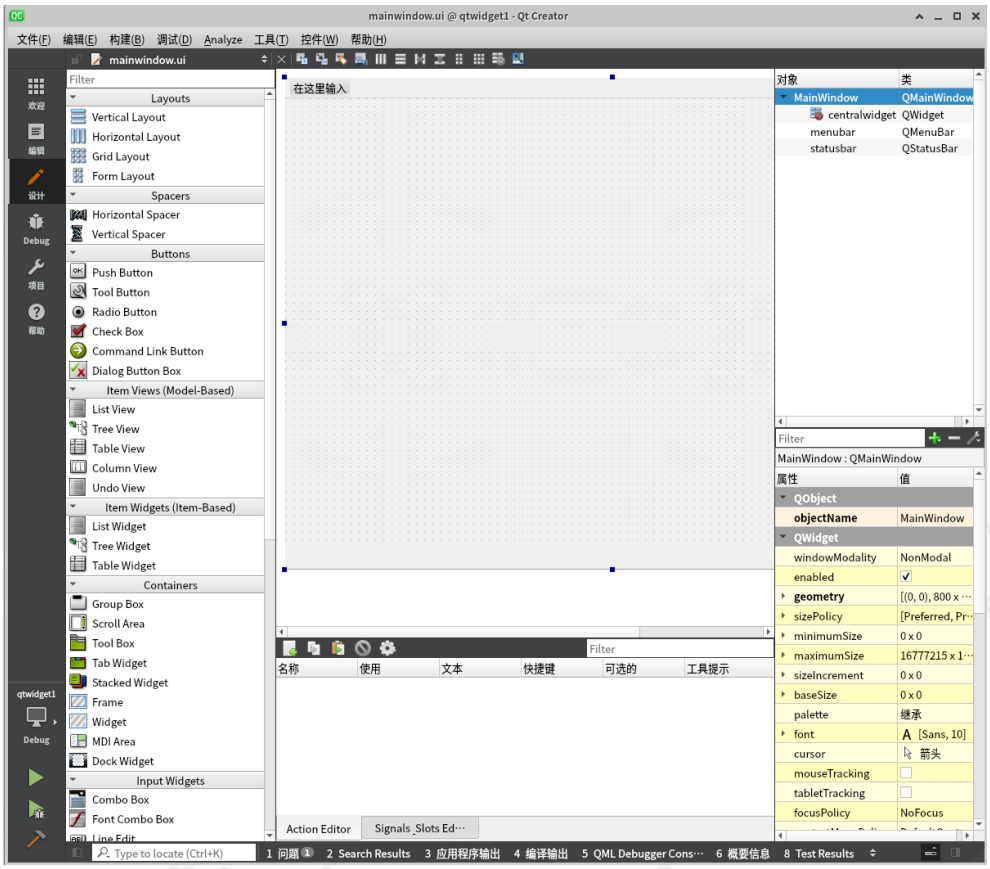

Qt Widgets Application — Qt Creator
Location
Build System
Details
Translation
Kits
Summary
Kit Selection
The following kits can be used for project qtwidget1:
Type to filter kits by name...
Select all kits
桌面
详情
Debug
/home/cek/CekApp/code/qt/build-qtwidget1-unknown-Debug
浏览...
Release
/home/cek/CekApp/code/qt/build-qtwidget1-unknown-Release
浏览...
Profile
/home/cek/CekApp/code/qt/build-qtwidget1-unknown-Profile
浏览...
< 上一步(B)
下一步(N) >
取消



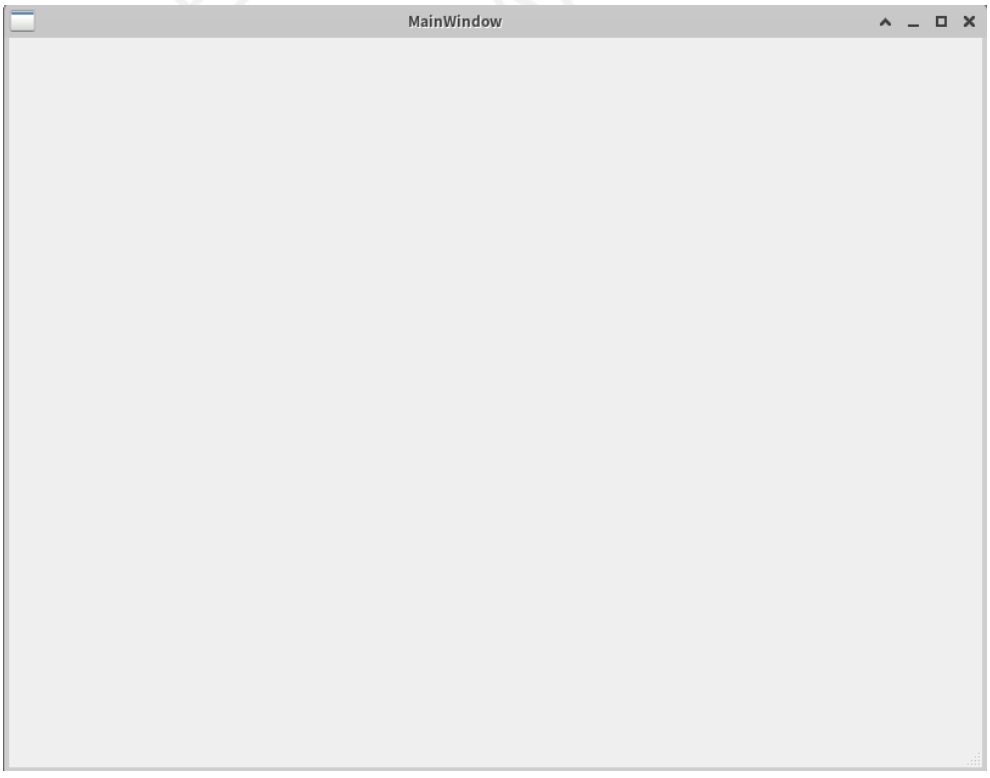
（3）创建完毕之后会进入项目源文件界面：



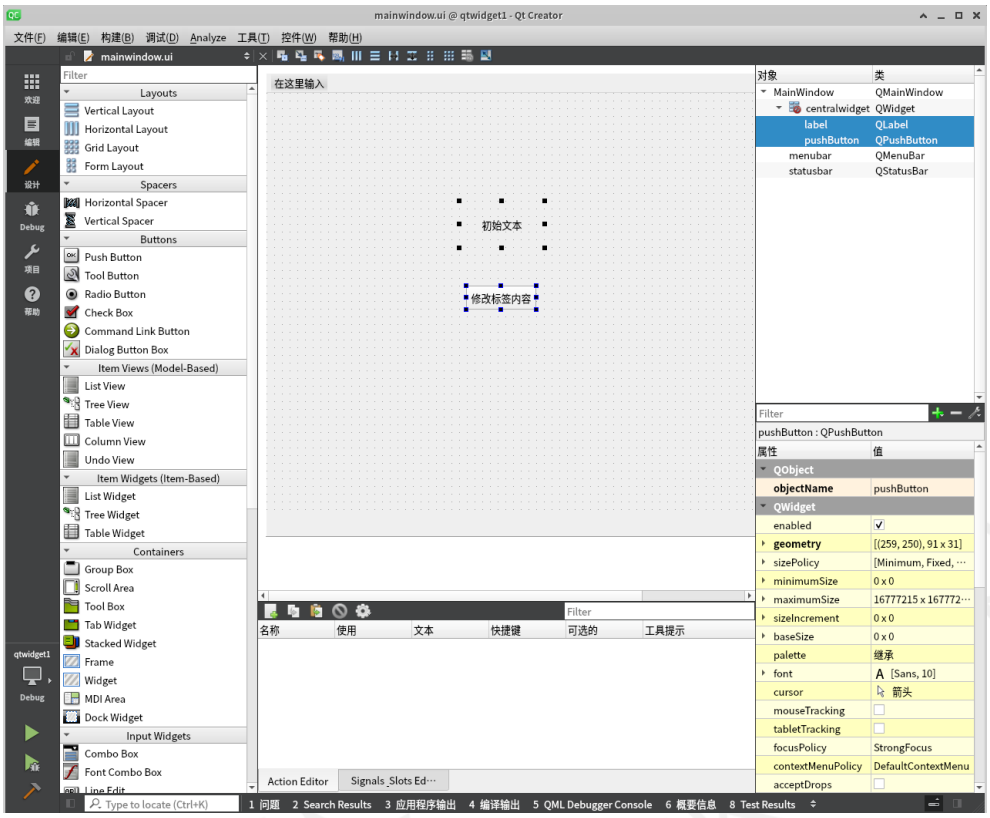
（4）双击“mainwindow.ui”文件进入 ui 资源文件编辑界面：



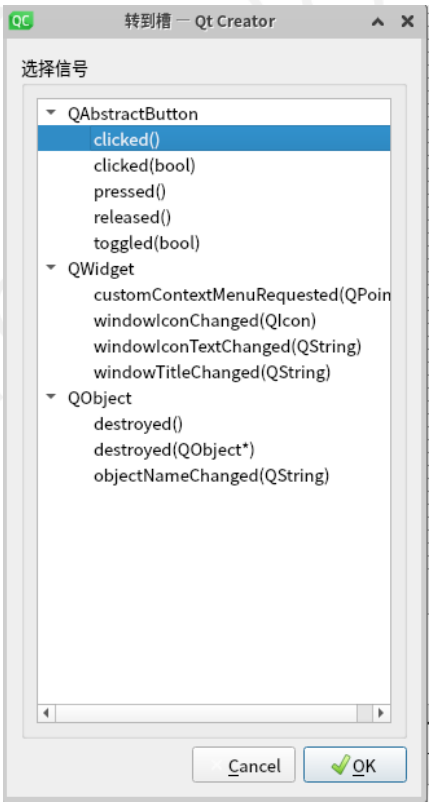
（5）点击“运行”图标，编译运行确认开发环境是否正常：



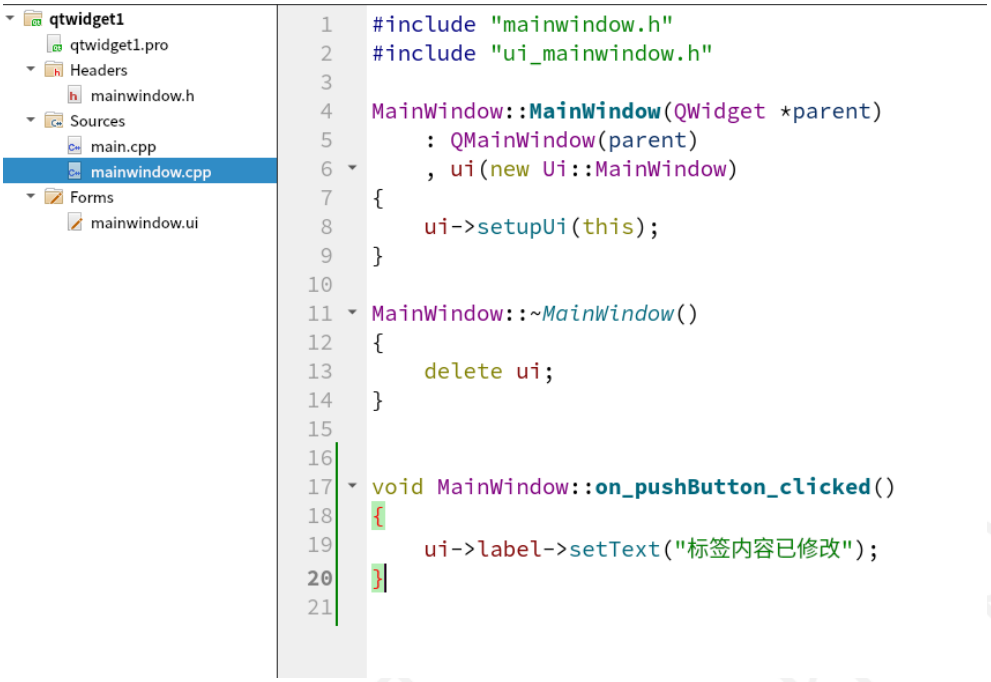
（6）在 ui 资源文件编辑界面添加控件，并调整布局：



（7）鼠标右键点击资源文件中的按钮，选择“转到槽...”：



（8）选择“clicked()”项跳转到对应 cpp 源码位置完善该按钮的鼠标单击事件：



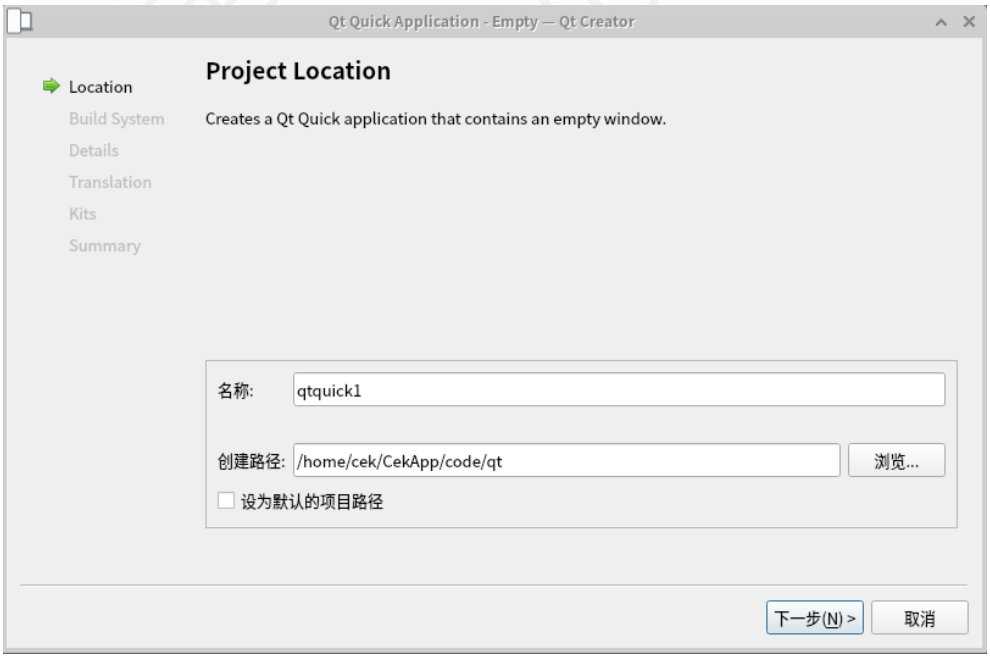

```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_pushButton_clicked()
{
    ui->label->setText("标签内容已修改");
}

```

（9）编写完成之后保存并编译运行查看效果：

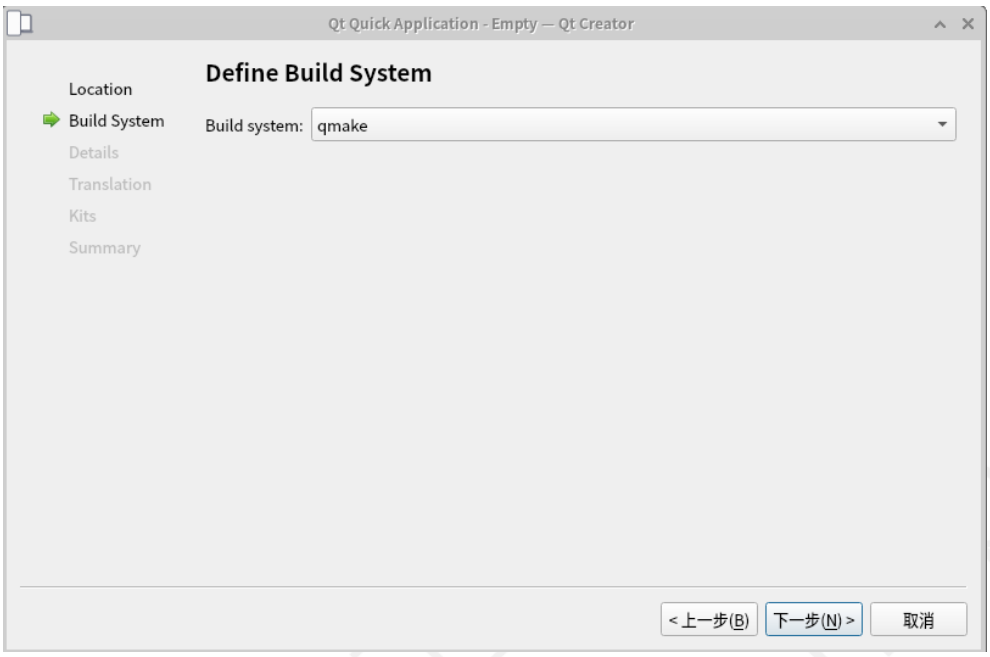
（10）点击按钮确认代码效果：

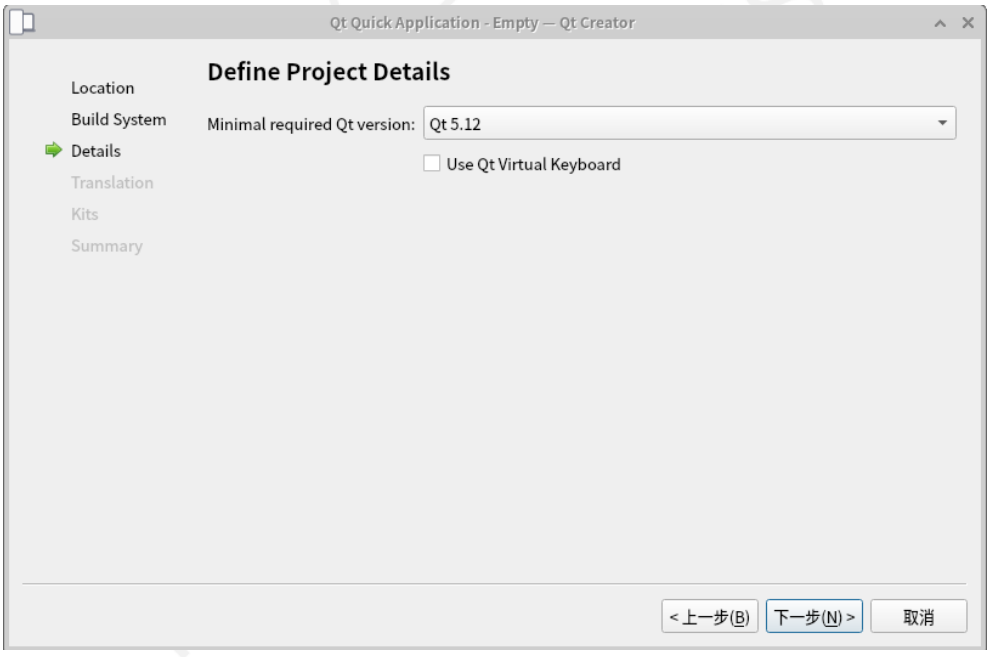
5.3.2．QtQuick 示例

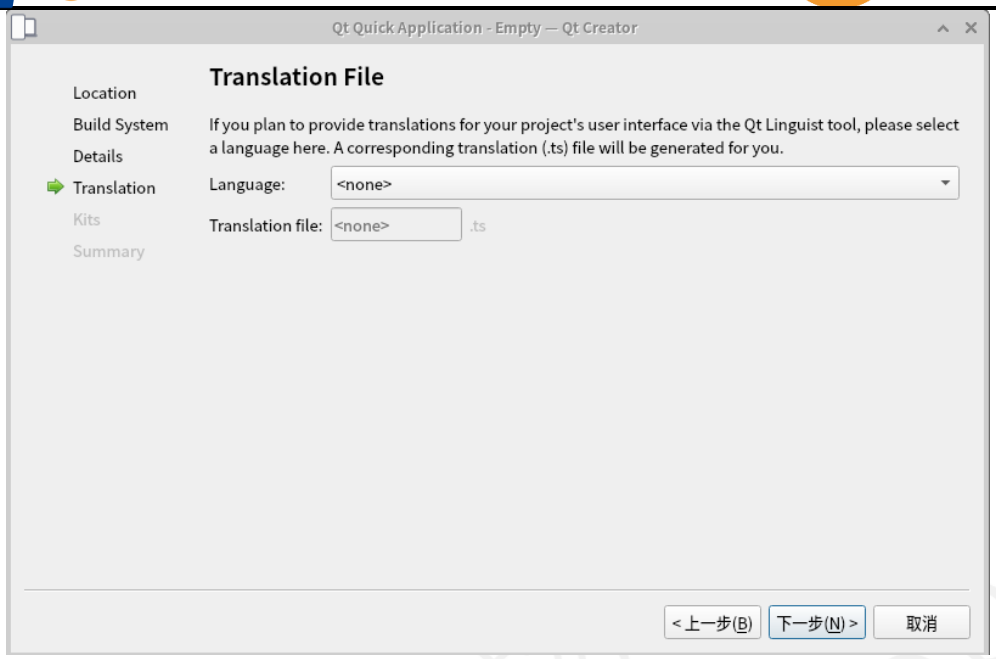
（1）在主页点击“New”新建工程：

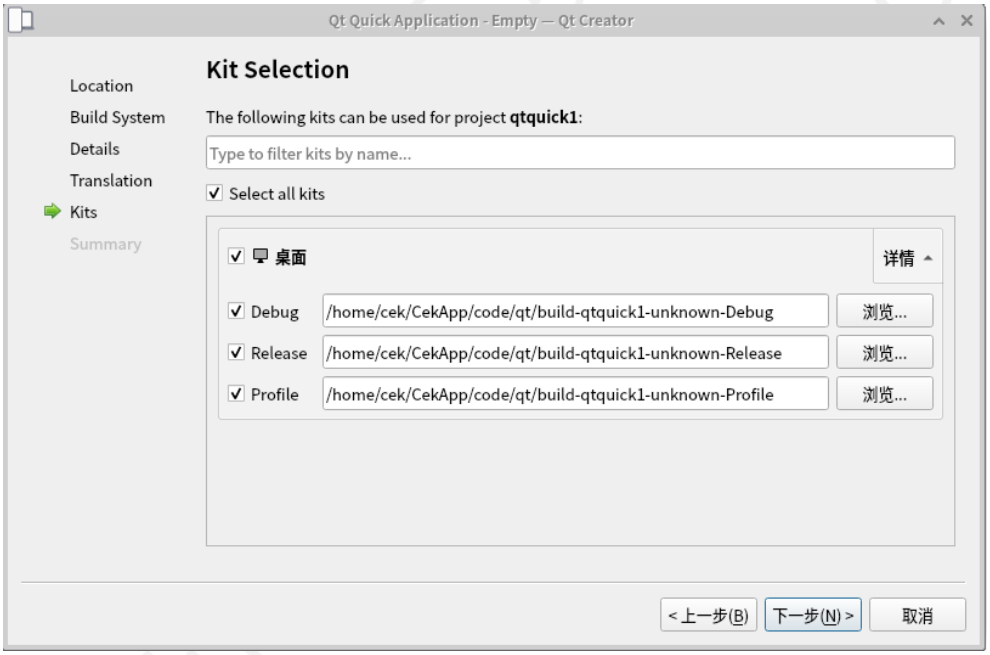


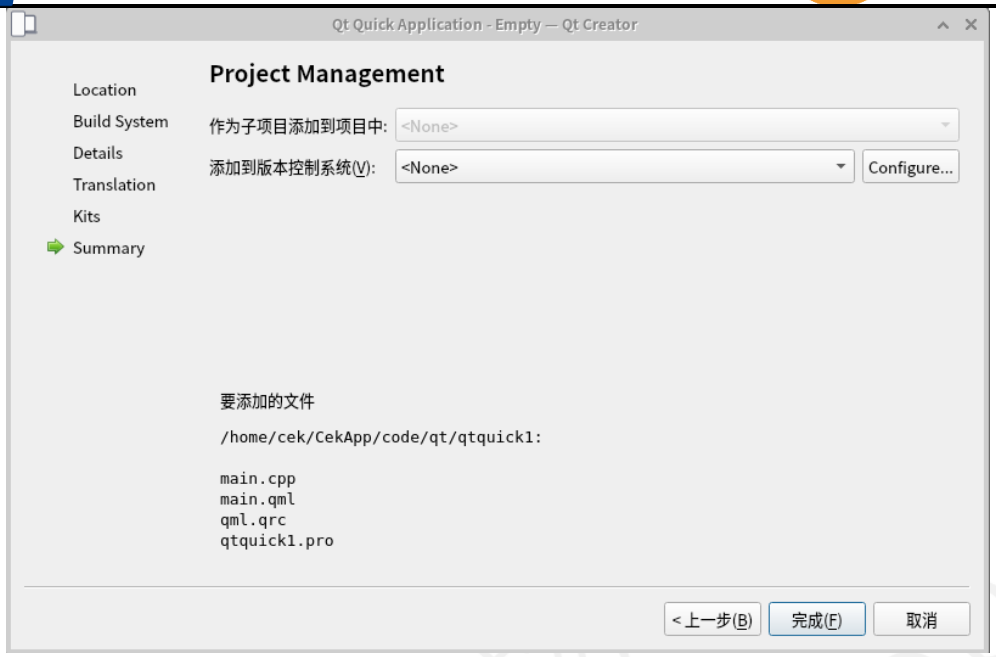
（2）选择“Qt Quick Application - Empty”创建 QtQuick 桌面应用的空模板，按默认配置依次点击“下一步”创建：



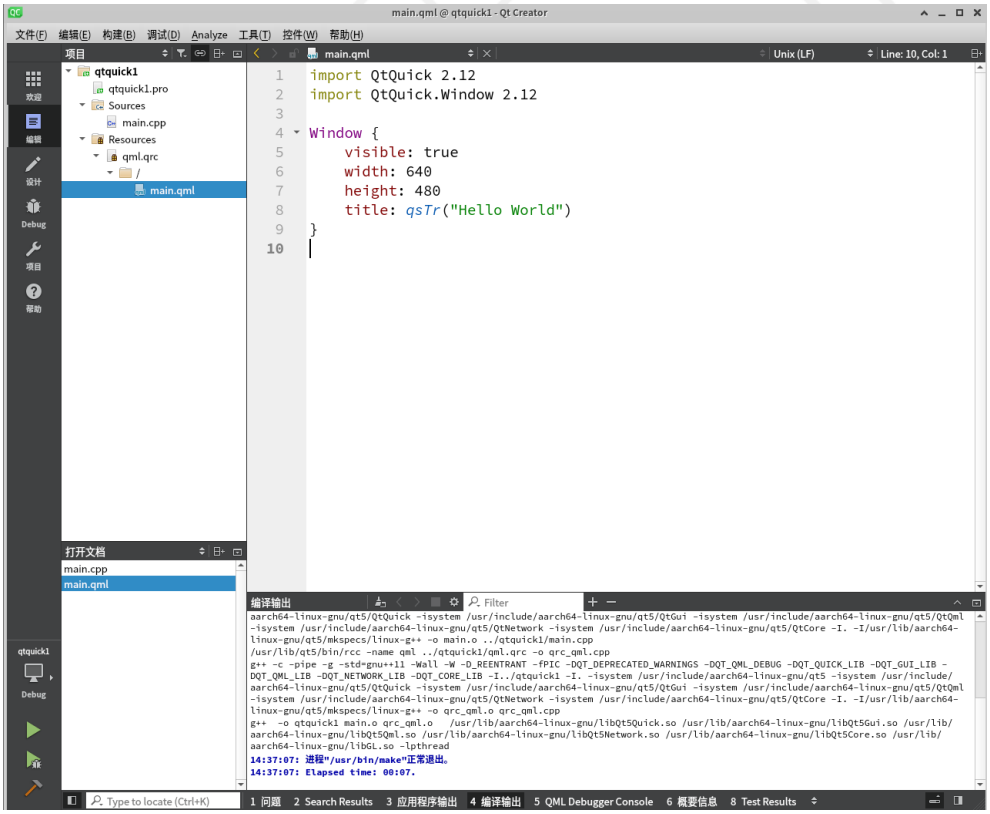



Qt Quick Application - Empty — Qt Creator
Location
Build System
Details
Translation
Kits
Summary
Translation File
If you plan to provide translations for your project's user interface via the Qt Linguist tool, please select a language here. A corresponding translation (.ts) file will be generated for you.
Language:
<none>
Translation file:
<none>
.ts
< 上一步(B)
下一步(N) >
取消

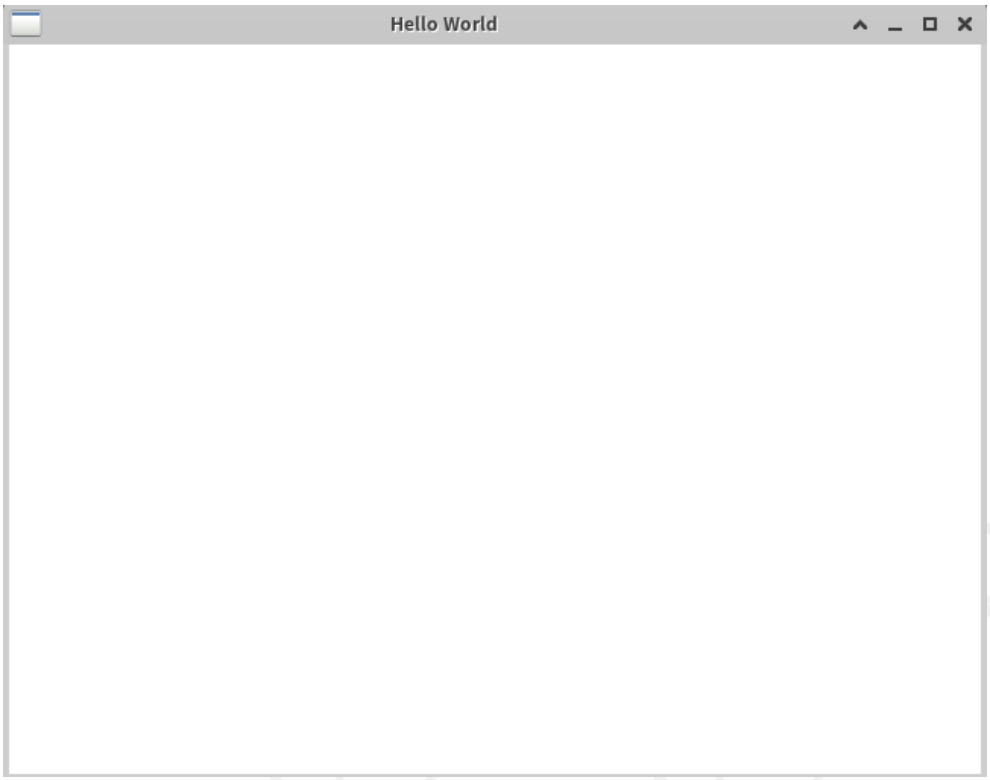

Qt Quick Application - Empty — Qt Creator
Location
Build System
Details
Translation
Kits
Summary
Kit Selection
The following kits can be used for project qtquick1:
Type to filter kits by name...
Select all kits
桌面
详情
Debug
/home/cek/CekApp/code/qt/build-qtquick1-unknown-Debug
浏览...
Release
/home/cek/CekApp/code/qt/build-qtquick1-unknown-Release
浏览...
Profile
/home/cek/CekApp/code/qt/build-qtquick1-unknown-Profile
浏览...
< 上一步(B)
下一步(N) >
取消



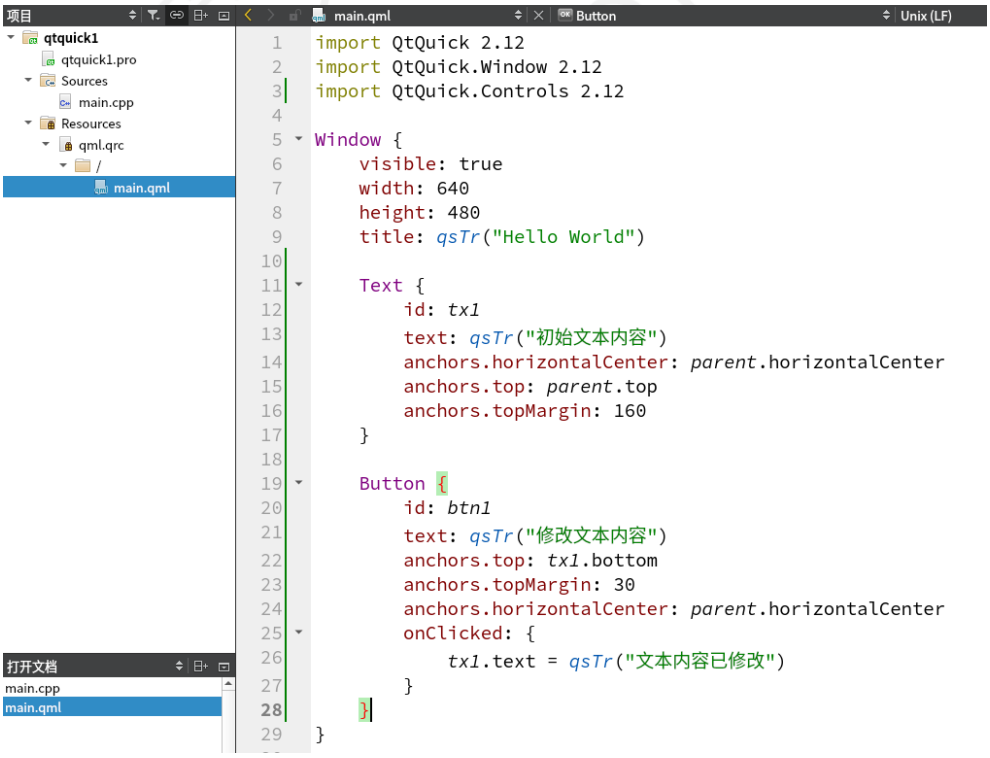
（3）创建完毕之后会进入项目源文件界面：



（4）创建完毕之后会进入项目源文件界面：



（5）修改源码，在 qml 文件中添加相关控件并编写控制代码：



（6）编写完成之后保存并编译运行查看效果：

（7）点击按钮确认代码执行效果：

6．高手进阶

用户在编译前，需安装好交叉编译需要的通用工具，以下只对针对飞腾派的编译进行介绍，所有操作位于 PC 端。

6.1．交叉编译环境搭建

把 gcc-linaro-7.5.0-2019.12-x86_64_aarch64 版本的交叉编译链解压到指定的一个目录，比如放在 /opt/toolchains/下：

```
zero@pc:~$ ls /opt/toolchains/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linuxgnu/
aarch64-linux-gnu bin gcc-linaro-7.5.0-2019.12-linux-manifest.txt include
lib libexec share
```

6.2．uboot 编译

（1）uboot 源码目录内容如下

```
zero@pc:~/work/u-boot-v1.40-pi$ ls
api configs include post u-boot.bin
u-boot-nodtb.bin
arch disk Kbuild README u-boot.cfg
u-boot-sd-boot-0511.bin
board doc Kconfig scripts u-boot.cfg.configs
u-boot-sd-boot-0605.bin
boot drivers lib set_env.sh u-boot.dtb
u-boot.srec
build_uboot.sh dts Licenses System.map u-boot-dtb.bin
u-boot.sym
cmd env MAINTAINERS test u-boot-flash-boot-0511.bin
common examples Makefile tools u-boot.lds
config.mk fs net u-boot u-boot.map
```

set_env.sh 和 ./build_uboot.sh 分别是编译环境配置脚本和编译脚本，用户可以根据需要修改。

set_env.sh 内容如下：

```
zero@pc:~/work/u-boot-v1.40-pi$ cat ./set_env.sh
export PATH="/opt/toolchains/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-
linux-gnu/bin/:$PATH"
export CROSS_COMPILE=aarch64-linux-gnu-
```

新增 PATH 的路径要根据用户放置交叉编译链的目录调整。

（2）使用方法

配置环境，新建的终端运行一次即可：

```
zero@pc:~/work/u-boot-v1.40-pi$ source set_env.sh
```

开始编译：

```
zero@pc:~/work/u-boot-v1.40-pi$ ./build_uboot.sh
```

（3）编译结果

uboot 镜像位于工程目录下：

```
zero@pc:~/work/u-boot-v1.40-pi$ ls u-boot.bin
u-boot.bin
```

6.3．kernel 编译

（1）内核源码目录内容如下：

```
zero@pc:~/work/phytium-linux-kernel-master-Uboot-V1.0.0$ ls
arch CREDITS include lib modules.builtin
samples tools
block crypto init LICENSES modules.order
scripts usr
build_kernel.sh Documentation ipc MAINTAINERS Module.symvers
security virt
built-in.a drivers Kbuild Makefile net
set_env.sh vmlinux
certs firmware Kconfig mm pack_module.sh sound
vmlinux.o
COPYING fs kernel modules README
System.map
```

set_env.sh 和 ./build_kernel.sh 分别是编译环境配置脚本和编译脚本，用户可以根据需要修改。

set_env.sh 内容如下：

```
zero@pc:~/work/phytium-linux-kernel-master-Uboot-V1.0.0$ cat
set_env.sh
export PATH=$PATH:"/opt/toolchains/gcc-linaro-7.5.0-2019.12-
x86_64_aarch64-linux-gnu/bin"
export CROSS_COMPILE=aarch64-linux-gnuexport ARCH=arm64
```

新增 PATH 的路径要根据用户放置交叉编译链的目录调整。

（2）使用方法

配置环境，新建的终端运行一次即可：

```
zero@pc:~/work/phytium-linux-kernel-master-Uboot-V1.0.0$ source
set_env.sh
```

开始编译

```
zero@pc:~/work0/phytium-linux-kernel-master-Uboot-V1.0.0$
./build_kernel.sh
```

（3）编译结果

内核镜像位于：

```
zero@pc:~/work/phytium-linux-kernel-master-Uboot-V1.0.0$ ls
./arch/arm64/boot/Image
./arch/arm64/boot/Image
```

设备树位于：

```
zero@pc:~/work/phytium-linux-kernel-master-Uboot-V1.0.0$ ls
./arch/arm64/boot/dts/phytium/phytium-pi-board.dtb
./arch/arm64/boot/dts/phytium/phytium-pi-board.dtb
```

module 位于：

```
zero@pc:~/work/phytium-linux-kernel-master-Uboot-V1.0.0$ ls
./modules/lib/modules/
4.19.246-phytium-embeded+
```

6.4．rootfs 制作

6.4.1．准备环境和工具

（1）安装相关工具

安装 debootstrap 与 qemu-user-static：

```
$ sudo apt-get install apt-transport-https qemu qemu-user-static binfmt-support debootstrap
```

（2）准备切换脚本

制作过程中需要切换到目标根文件系统的环境中进行操作，使用到的脚本文件 ch-mount.sh 内容如下：

```
$ cat ch-mount.sh
#!/bin/bash
function mnt() {
    echo "MOUNTING"
```

```
        sudo mount -t proc /proc ${2}proc
        sudo mount -t sysfs /sys ${2}sys
        sudo mount -o bind /dev ${2}dev
        sudo mount -o bind /dev/pts ${2}dev/pts
        sudo chroot ${2}
}
function umnt(){
        echo "UNMOUNTING"
        sudo umount ${2}proc
        sudo umount ${2}sys
        sudo umount ${2}dev/pts
        sudo umount ${2}dev
}
if [ "$1" == "-m" ] && [ -n "$2" ] ;
then
        mnt $1 $2
elif [ "$1" == "-u" ] && [ -n "$2" ];
then
        umnt $1 $2
else
        echo ""
        echo "Either 1'st, 2'nd or bothparameters were missing"
        echo ""
        echo "1'st parameter can be one ofthese: -m(mount) OR -u(umount)"
        echo "2'nd parameter is the full pathof rootfs directory(with trailing '/')"
        echo ""
        echo "For example: ch-mount -m/media/sdcard/"
        echo ""
        echo 1st parameter : ${1}
        echo 2nd parameter : ${2}
```

6.4.2．Ubuntu20.04 根文件系统制作

（1）生成基础根文件系统

使用下面的命令通过 debootstrap 工具生成 arm64 基础 rootfs 包：

```
$ sudo debootstrap --arch=arm64 focal rootfs http://mirrors.aliyun.com/ubuntu-ports/
```

```
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ sudo debootstrap --arch=arm64 focal rootfs http://mirrors.aliyun.com/ubuntu-ports/
I: Retrieving InRelease
I: Checking Release signature
I: Valid Release signature (key id F6ECB3762474EDA9D21B7022871920D1991BC93C)
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
I: Checking component main on http://mirrors.aliyun.com/ubuntu-ports...
I: Retrieving adduser 3.118ubuntu2
I: Validating adduser 3.118ubuntu2
I: Retrieving apt 2.0.2
I: Validating apt 2.0.2
I: Retrieving apt-utils 2.0.2
I: Validating apt-utils 2.0.2
I: Retrieving base-files 11ubuntu5
I: Validating base-files 11ubuntu5
I: Retrieving base-passwd 3.5.47
I: Validating base-passwd 3.5.47
I: Retrieving bash 5.0-6ubuntu1
I: Validating bash 5.0-6ubuntu1
I: Retrieving bsdutils 1:2.34-0.1ubuntu9
I: Validating bsdutils 1:2.34-0.1ubuntu9
I: Retrieving bzip2 1.0.8-2
I: Validating bzip2 1.0.8-2
I: Retrieving ca-certificates 20190110ubuntu1
I: Validating ca-certificates 20190110ubuntu1
I: Retrieving console-setup 1.194ubuntu3
I: Validating console-setup 1.194ubuntu3
I: Retrieving console-setup-linux 1.194ubuntu3
I: Validating console-setup-linux 1.194ubuntu3
I: Retrieving coreutils 8.30-3ubuntu2
I: Validating coreutils 8.30-3ubuntu2
I: Retrieving cron 3.0pl1-136ubuntu1
I: Validating cron 3.0pl1-136ubuntu1
I: Retrieving dash 0.5.10.2-6
I: Validating dash 0.5.10.2-6
I: Retrieving dbus 1.12.16-2ubuntu2
I: Validating dbus 1.12.16-2ubuntu2
I: Retrieving debconf 1.5.73
I: Validating debconf 1.5.73
I: Retrieving debconf-i18n 1.5.73
I: Validating debconf-i18n 1.5.73
I: Retrieving debianutils 4.9.1
I: Validating debianutils 4.9.1
I: Retrieving diffutils 1:3.7-3
```

等待命令执行完毕之后会在当前目录下生成一个 rootfs 文件夹，里面包含了 arm64 版本的 Ubuntu20.04 基础 rootfs 包：

```
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ ls rootfs/
bin  boot  dev  etc  home  lib  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ sudo du -sh rootfs/
328M    rootfs/
cecport@F10101124X00849-W:base$
```

（2）配置基础软件

①更换软件源

可以直接在主机中使用编辑工具修改目标 rootfs 的软件源：

```
$ sudo debootstrap --arch=arm64 focal rootfs http://mirrors.aliyun.com/ubuntu-ports/
```

将其中的内容替换为如下内容：

```
deb https://mirrors.aliyun.com/ubuntu-ports/ focal main restricted universe multiverse
deb https://mirrors.aliyun.com/ubuntu-ports/ focal-updates main restricted universe multiverse
deb https://mirrors.aliyun.com/ubuntu-ports/ focal-backports main restricted universe multiverse
deb https://mirrors.aliyun.com/ubuntu-ports/ focal-security main restricted universe multiverse
```

②准备 qemu

将相应的 qemu 文件拷贝至目标 rootfs 中：

```
$ sudo cp /usr/bin/qemu-aarch64-static rootfs/usr/bin/
```

③操作环境切换

使用前面制作的 ch-mount.sh 切换至模拟环境中配置目标 rootfs 更加直观。

切换至模拟操作环境命令：

```
$ ./ch-mount.sh -m rootfs/
```

退出模拟操作环境：

```
$ ./ch-mount.sh -u rootfs/
```

```
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ ./ch-mount.sh -m rootfs/
MOUNTING
root@F10101124X00849-W:/#
root@F10101124X00849-W:/#
root@F10101124X00849-W:/# ls
bin  boot  dev  etc  home  lib  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@F10101124X00849-W:/#
root@F10101124X00849-W:/#
root@F10101124X00849-W:/# exit
exit
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ ./ch-mount.sh -u rootfs/
UNMOUNTING
cecport@F10101124X00849-W:base$
```

切换环境之后，可以直接使用默认安装的相关工具与命令，命令操作会直接影响目标 rootfs。

④更新软件

首次进入目标 rootfs 模拟环境中进行操作建议先更新默认安装的软件：

```
/# apt-get update && apt-get upgrade
```

安装自动补全工具：

```
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ ./ch-mount.sh -m rootfs/
MOUNTING
root@F10101124X00849-W:/#
root@F10101124X00849-W:/#
root@F10101124X00849-W:/# apt-get update && apt-get upgrade
Hit:1 https://mirrors.aliyun.com/ubuntu-ports focal InRelease
Get:2 https://mirrors.aliyun.com/ubuntu-ports focal-updates InRelease [114 kB]
Get:3 https://mirrors.aliyun.com/ubuntu-ports focal-backports InRelease [108 kB]
Get:4 https://mirrors.aliyun.com/ubuntu-ports focal-security InRelease [114 kB]
Get:5 https://mirrors.aliyun.com/ubuntu-ports focal/main Translation-en [506 kB]
Get:6 https://mirrors.aliyun.com/ubuntu-ports focal/restricted arm64 Packages [1300 B]
Get:7 https://mirrors.aliyun.com/ubuntu-ports focal/restricted Translation-en [6212 B]
Get:8 https://mirrors.aliyun.com/ubuntu-ports focal/universe arm64 Packages [8458 kB]
Get:9 https://mirrors.aliyun.com/ubuntu-ports focal/universe Translation-en [5124 kB]
Get:10 https://mirrors.aliyun.com/ubuntu-ports focal/multiverse arm64 Packages [114 kB]
Get:11 https://mirrors.aliyun.com/ubuntu-ports focal/multiverse Translation-en [104 kB]
Get:12 https://mirrors.aliyun.com/ubuntu-ports focal-updates/main arm64 Packages [1928 kB]
Get:13 https://mirrors.aliyun.com/ubuntu-ports focal-updates/main Translation-en [440 kB]
```

Ubuntu 系统的终端命令自动补全功能是由 bash-completion 实现的，可以在模拟环境中

直接通过 apt 来安装它：

```
/# apt-get install bash-completion
```

安装完成之后在模拟环境中的输入命令时可以通过 tab 键进行自动补全，如果未生效需要检查/etc/profile 等相关配置文件中是否屏蔽了相关功能。

⑤安装 vim

vim 是 Linux 系统中最常用的文本编辑工具，可以在模拟环境中进行安装：

```
/# apt-get install vim
```

⑥配置以太网

较新版本的 Ubuntu 系统默认通过 netplan 管理网络，需要自行调整相应的配置文件。对于当前板卡，板端硬件有设计 2 个以太网口(eth0 与 eth1)，相应的配置文件可以参考下面的内容进行配置：

```
/# vim /etc/netplan/01-network-all.yaml
network:
  ethernets:
    eth0:
      dhcp4: yes
    eth1:
      dhcp4: yes
  version: 2
  renderer: NetworkManager
```

```
root@F10101124X00849-W:/#
root@F10101124X00849-W:/# vim /etc/netplan/01-network-all.yaml
root@F10101124X00849-W:/#
root@F10101124X00849-W:/# cat /etc/netplan/01-network-all.yaml
network:
  ethernets:
    eth0:
      dhcp4: yes
    eth1:
      dhcp4: yes
  version: 2
  renderer: NetworkManager
root@F10101124X00849-W:/#
```

其中最后一行配置“ renderer: NetworkManager”是可选的，代表是否关联 Ubuntu 的 NetworkManager，通常与桌面的网络管理界面相关。

⑦配置系统用户

修改 root 用户密码：

```
/# passwd root
```

新增 user 用户：

```
/# adduser cek
```

```
root@F10101124X00849-W:/#
root@F10101124X00849-W:/# passwd root
New password:
Retype new password:
passwd: password updated successfully
root@F10101124X00849-W:/#
root@F10101124X00849-W:/#
root@F10101124X00849-W:/# adduser user
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = "en_US",
        LC_ALL = (unset),
        LC_NUMERIC = "zh_CN.UTF-8",
        LC_TIME = "zh_CN.UTF-8",
        LC_MEASUREMENT = "zh_CN.UTF-8",
        LC_TELEPHONE = "zh_CN.UTF-8",
        LC_IDENTIFICATION = "zh_CN.UTF-8",
        LC_PAPER = "zh_CN.UTF-8",
        LC_MONETARY = "zh_CN.UTF-8",
        LC_NAME = "zh_CN.UTF-8",
        LC_ADDRESS = "zh_CN.UTF-8",
        LANG = "en_US.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
Adding user `user' ...
Adding new group `user' (1000) ...
Adding new user `user' (1000) with group `user' ...
Creating home directory `/home/user' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for user
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@F10101124X00849-W:/#
```

调整 user 用户权限：

```
/# vim /etc/sudoers
```

新增行：

```
user    ALL=(ALL:ALL) ALL
```

```
#
Defaults    env_reset
Defaults    mail_badpass
Defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
user    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

修改完成之后保存退出，当前的 rootfs 可以作为一个不带桌面的基础版本 Ubuntu20.04 根文件系统。

（3）安装与配置桌面

①安装 xfce4

xfce4 是基于 X11 的轻量级桌面，在 Ubuntu20.04 上可以直接通过 apt 进行安装。通过自动提示功能，可以发现 xfce4 在 apt 中的组件非常丰富：

```
user@F10101124X00849-W:~$
user@F10101124X00849-W:~$ sudo apt-get install xfce4
xfce4                       xfce4-fsguard-plugin        xfce4-places-plugin         xfce4-systemload-plugin
xfce4-appfinder             xfce4-genmon-plugin         xfce4-power-manager         xfce4-taskmanager
xfce4-appmenu-plugin        xfce4-goodies               xfce4-power-manager-data    xfce4-terminal
xfce4-battery-plugin        xfce4-indicator-plugin      xfce4-power-manager-plugins xfce4-time-out-plugin
xfce4-clipman               xfce4-mailwatch-plugin      xfce4-pulseaudio-plugin     xfce4-timer-plugin
xfce4-clipman-plugin        xfce4-mount-plugin          xfce4-screensaver           xfce4-verve-plugin
xfce4-cpufreq-plugin        xfce4-mpc-plugin            xfce4-screenshooter         xfce4-volumed
xfce4-cpugraph-plugin       xfce4-netload-plugin        xfce4-sensors-plugin        xfce4-wavelan-plugin
xfce4-datetime-plugin       xfce4-notes                 xfce4-session               xfce4-weather-plugin
xfce4-dev-tools             xfce4-notes-plugin          xfce4-settings              xfce4-whiskermenu-plugin
xfce4-dict                  xfce4-notifyd               xfce4-smartbookmark-plugin  xfce4-xkb-plugin
xfce4-diskperf-plugin       xfce4-panel                 xfce4-sntray-plugin
xfce4-equake-plugin         xfce4-panel-dev             xfce4-sntray-plugin-common
xfce4-eyes-plugin           xfce4-panel-profiles        xfce4-statusnotifier-plugin
```

建议安装 xfce4 与 xfce4-goodies 即可：

```
~$ sudo apt-get install xfce4 xfce4-goodies
```

如果在安装过程中弹出“Default display manager: ”相关的选项，推荐选择 lightdm。

②安装用户登录管理工具

Ubuntu 最常用的用户登录管理工具包括 lightdm 与 gdm，对于性能相对较弱的嵌入式设备，推荐使用 lightdm。如果在前面的步骤中有安装过 lightdm 则跳过此步。

在模拟环境的 user 用户中，使用下面的命令进行安装：

```
~$ sudo apt-get install lightdm
```

安装相应的登录界面：

```
~$ sudo apt-get install lightdm-gtk-greeter
```

③配置默认登录会话

在配置的 Ubuntu 系统中，所有的 X11 桌面会话配置文件位于 /usr/share/xsessions/ 路径，通过下面的命令查看当前系统中所包含的桌面会话：

```
~$ ls /usr/share/xsessions/
```

```
user@F10101124X00849-W:~$ ls /usr/share/xsessions/
ubuntu.desktop  xfce.desktop
```

lightdm 的配置文件位于 /usr/share/lightdm/lightdm.conf.d/ 路径下，默认配置启动的桌面会话是 gnome 桌面，可以通过下面的步骤将其调整为 xfce4 桌面：

```
~$ sudo cp /usr/share/lightdm/lightdm.conf.d/50-ubuntu.conf /usr/share/lightdm/lightdm.conf.d/60-xfce.conf
~$ sudo vim /usr/share/lightdm/lightdm.conf.d/60-xfce.conf
```

配置：user-session=xfce

```
user@F10101124X00849-W:~$ cat /usr/share/lightdm/lightdm.conf.d/60-xfce.conf
[Seat:*]
user-session=xfce
```

配置完成后按照前面介绍的方法退出模拟环境，烧录后首次运行时需要在登录界面手动选择 xfce 桌面，否则默认按照字母排序会启动 gnome 桌面。

6.4.3．烧录验证

（1）SD 卡分区

选择一张超过 8GB 的 SD 卡插入当前系统中，使用 fdisk 命令确认新插入的 SD 卡在系统中的具体节点：

```
cecport@F10101124X00849-W:SD卡启动$ sudo fdisk -l
Disk /dev/nvme0n1: 1.88 TiB, 2048408248320 bytes, 4000797360 sectors
Disk model: KXG6APNV2T04 TOSHIBA
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: E948F076-57FE-4050-99D1-5818179595E3

Device           Start        End    Sectors  Size Type
/dev/nvme0n1p1    2048    1050623    1048576  512M EFI System
/dev/nvme0n1p2 1050624 4000796671 3999746048  1.9T Linux filesystem


Disk /dev/sda: 57.98 GiB, 62226694144 bytes, 121536512 sectors
Disk model: Storage Device
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

通过上图可知新插入的 SD 卡为/dev/sda，下面通过 fdisk 对该设备进行分区：

```
cecport@F10101124X00849-W:SD卡启动$ sudo fdisk /dev/sda

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk /dev/sda: 57.98 GiB, 62226694144 bytes, 121536512 sectors
Disk model: Storage Device
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x702b74f4

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-121536511, default 2048): 131072
Last sector, +/-sectors or +/-size{K,M,G,T,P} (131072-121536511, default 121536511):

Created a new partition 1 of type 'Linux' and of size 57.9 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

cecport@F10101124X00849-W:SD卡启动$
cecport@F10101124X00849-W:SD卡启动$ ls /dev/sda*
/dev/sda  /dev/sda1
cecport@F10101124X00849-W:SD卡启动$
```

其中第一个分区需要跳过前 64M 的空间，前 64M 后续会被用于存放 uboot、kernel 以及 dtb 等内容。第一个分区的起始区块偏移为：64×1024×1024÷512=131072，其中区块默认大小为 512bytes。

（2）格式化目标分区

上一步分区操作之后的 SD 卡中的第一个分区用于存放根文件系统，需要使用 mkfs.ext4 命令将其格式化为 ext4 格式：

```
$ sudo mkfs.ext4 /dev/sda1
```

```
cecport@F10101124X00849-W:SD卡启动$ sudo mkfs.ext4 /dev/sda1
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 15175680 4k blocks and 3801088 inodes
Filesystem UUID: 2b869ea7-45e1-4f5d-923f-b367a80f8930
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424

Allocating group tables: done
Writing inode tables: done
Creating journal (65536 blocks):
done
Writing superblocks and filesystem accounting information: done

cecport@F10101124X00849-W:SD卡启动$
```

（3）拷贝根文件系统

在之前制作根文件系统的目录按照下面的命令将制作好的根文件系统拷贝至 SD 卡中的第一个分区：

```
$ mkdir tmp
$ sudo mount /dev/sda1 tmp/
$ sudo cp -rfp rootfs/* tmp/
$ sudo umount tmp
```

```
cecport@F10101124X00849-W:base$ mkdir tmp
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ sudo mount /dev/sda1 tmp/
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ sudo cp -rfp rootfs/* tmp/
cecport@F10101124X00849-W:base$
cecport@F10101124X00849-W:base$ sudo umount tmp
```

（4）写入 boot

对于完全 SD 卡启动的情况，需要将 uboot、kernel、dtb 等内容打包的 bin 文件写入 SD 卡前 64M 的空间中，这个写入过程会破坏 SD 卡本身的分区表，在写入之前可以通过下面的方式保存 SD 卡的分区表内容：

```
$ sudo dd if=/dev/sda of=table-4M.bin bs=1024 count=4096
```

后续如果需要复原分区表结构，使用下面的命令回写即可：

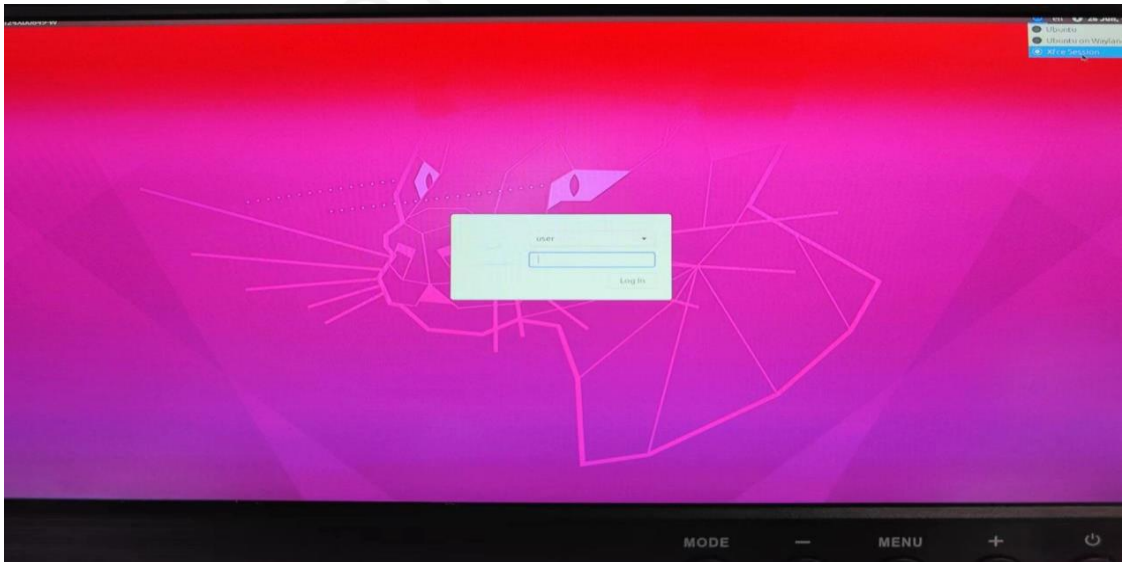
```
$ sudo dd if=table-4M.bin of=/dev/sda
```

将准备的 boot 打包文件写入 SD 卡：

```
$ sudo dd if=fip-all-sd-boot.bin of=/dev/sda
$ sync
```

（5）SD 卡写入

将前面制作好的 SD 卡插入板端，首次上电启动系统进入登录界面时需要手动在右上角选择桌面会话为：Xfce Session：



6.5．linux 驱动开发

拓展知识篇

7．拓展应用

扩展应用详情，请查看官方提供的相关应用手册。

扩展应用名称	是否支持	查看地址/链接
openCV	支持	用户手册目录下
4G 通信	支持	用户手册目录下
5G 通信	支持	用户手册目录下
AI 加速卡	支持	用户手册目录下
持续更新中		

8．注意事项

8.1．常规事项

在使用过程中，为避免故障或损坏本产品，请注意下面（且不限于）问题点：

- ✧ 请仔细阅读说明书、注意事项等，确定开发板工作电压、工作电流，确定外设版本与接口类型，确定串口工作电平等
- ✧ 此产品不可超频使用，否则不予保修
- ✧ 请勿将此产品暴露于任何热源，此产品仅适合在正常室温中使用，以确保可靠运行
- ✧ 在通风良好的环境中运行此产品，在使用过程中请勿覆盖。根据开发板工作需要，判断散热情况，适当的为主芯片安装散热片、散热风扇
- ✧ 请注意开发板所有暴露的电极是否有短路可能。开发板底部严禁与金属或其他导体的接触
- ✧ 搬运时要小心，避免对电路板和连接器造成机械或电气损坏
- ✧ 请不要用手触摸开发板上的元器件，人体的静电容易损坏元件
- ✧ 不要用 IO 口直接驱动感性负载（电机/电磁阀/继电器等有线圈的负载），因为感性负载在断开的一瞬间会产生很高的反电动势，直接把 IO 口烧坏
- ✧ 请妥善保存包装及其它配件，以便保存。长期不用要注意防潮，防尘
- ✧ 产品的温湿度要求如下

参数	最佳范围	备注
工作温度	0~50℃	禁止超出范围使用
保存温度	20~30℃	
保存湿度	45~65%	相对湿度

8.2．附注

（1）为下列情况之一的产品，不实行免费保修：

- 超过保修服务期
- 无有效购买单据
- 进液、受潮或发霉
- 由于购买后跌落、强烈震动或擅自改动、误操作等非产品质量原因引起的故障和损坏
- 因为不可抗力造成损坏

（2）我公司保留所有产品中自主开发的相关软、硬件技术资料的知识产权；用户仅能将它们作为教学、实验、科研使用，不得从事任何商业用途，也不能将它们在网络上散发，或者通过截取、修改等方式来篡改它们的著作权

（3）本产品接受客户批量订购，公司将提供全方面的技术支持和服务

9．附录及常见问题解答