悟空派H3 Zero 用户手册

目 录

1. 悟空派H3 Zero的基本特性

1.1. 什么是悟空派H3 Zero

悟空派是一款开源的单板卡片电脑，新一代的Linux开发板，它可以运行Linux、Ubuntu 和 Debian 等操作系统。悟空派开发板（悟空派H3 Zero）使用全志 H3 系统级芯片，同时拥有 256MB/512MB DDR3 内存

2. 开发板使用介绍

2.1. 准备需要的配件

1) TF 卡，最小 8GB 容量的 class10 级以上的高速卡，建议使用闪迪的 TF 卡，WuKong Pi 测试都是使用闪迪的 TF 卡，其他牌子的 TF 卡可能会出现系统无法启动的问题

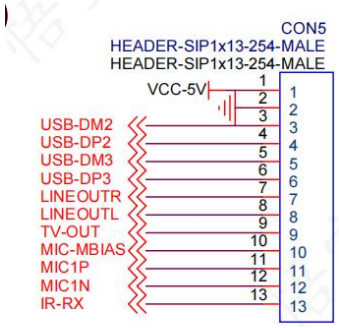
2) TF 卡读卡器，用于读写 TF 卡

3) 电源适配器，至少 5V/2A 的高品质 USB Type-C接口的电源适配器

4.悟空派H3 Zero 开发板上的 13pin 排针可以接上转接板来扩展开发板上没有的功能，转接板包含的功能有（此转接板需要自己设计）

1	麦克风	用于录音功能
2	模拟音视频输出接口	可用于接耳机播放音乐，或者通过 AV 线接电视输 出模拟音视频信号（仅安卓系统）
3	USB2.0 x 2	用于接 USB 键盘、鼠标以及 USB 存储设备
4	红外接收功能	通过红外遥控可以控制 Android 系统

5. 悟空派H3 Zero 13pin 排针的原理图如下所示



5) USB接口的鼠标和键盘，只要是标准USB接口的鼠标和键盘都可以，鼠标和键盘可以用来控制WuKong Pi开发板(**3/4/5/6脚 支持两路USB2.0接口**)

6) 红外遥控脚，主要用于控制安卓系统（**13脚**）

7) 百兆或者千兆网线，用于将开发板连接到因特网

8) AV视频线，由于开发板没有 HDMI 接口，如果想查看 Android 系统的视频输出， 需要通过 AV 视频线将开发板连接到电视（**7/8/9/10/11/12脚**）

10) 安装有 Ubuntu 和 Windows 操作系统的个人电脑

1	Ubuntu14.04 PC	可选，用于编译 Android 源码
2	Ubuntu18.04 PC	可选，用于编译 Linux 源码
3	Windows PC	用于烧录 Android 和 Linux 镜像

2.2. 下载开发板的镜像和相关的资料

1) armbian镜像下载网址为

https://pan.baidu.com/s/1scMeadmTEeOpsAw2o_tCJg?pwd=abcd

2)

a. armbian **源码**：保存在 github 上，链接地址为

https://github.com/Timfu2019/wukongpi_build

b. **用户手册和原理图：**芯片相关的数据手册也会放在这里

c. **官方工具：**主要包括开发板使用过程中需要用到的软件

2.3. 基于 Windows PC 将 Linux 镜像烧写到 TF 卡的方法

1) 首先准备一张 8GB或更大容量的TF卡，TF卡的传输速度必须为**class10** 以上，建 议使用闪迪等品牌的TF卡，格式化TF卡

2) 从下载的镜像解压后的文件中，以“**.img**”结尾的文件就是操作系统的镜像文件，大小一般都在 1GB以上

3) 使用**Win32Diskimager**烧录Linux镜像到TF卡

a. Win32Diskimager的下载页面为

```
http://sourceforge.net/projects/win32diskimager/files/Archive/
```

b. 下载完后直接安装即可，Win32Diskimager界面如下所示

a) 首先选择镜像文件的路径

b) 然后确认下TF卡的盘符和“**设备**”一栏中显示的一致

c) 最后点击“**写入**”即可开始烧录

c. 镜像写入完成后，点击“退出”按钮退出即可，然后就可以拔出TF卡插到开 发板中启动

2.4. 基于 Ubuntu PC 将 Linux 镜像烧写到 TF 卡的方法

1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 以上，建议使用闪迪等品牌的 TF 卡

2) 然后使用读卡器把 TF 卡插入电脑

3) 下载 balenaEtcher 软件，下载地址为

```
https://www.balena.io/etcher/
```

4) 进入 balenaEtcher 下载页面后，请通过下拉框选择 Linux 版本的软件进行下载

5) 下载完后使用 **unzip** 进行解压，解压后的 **balenaEtcher-1.5.109-x64.AppImage** 就是烧录需要的软件

```
test@test:~$ unzip balena-etcher-electron-1.5.109-linux-x64.zip
Archive:        balena-etcher-electron- 1.5. 109-linux-x64.zip
  inflating: balenaEtcher- 1.5. 109-x64.AppImage
test@test:~$ ls
balenaEtcher-1.5.109-x64.AppImage        balena-etcher-electron- 1.5. 109-linux-x64.zip
```

6) 从 WuKong Pi 的资料下载页面下载想要烧录的 Linux 操作系统镜像文件压缩包，然后使用解压软件解压，解压后的文件中，以“**.img**”结尾的文件就是操作系统的 镜像文件，大小一般在 1GB 以上

a. 7z 结尾的压缩包的解压命令如下所示

```
test@test:~$ 7z x       image_filename.7z
```

b. tar.gz 结尾的压缩包的解压命令如下所示

```
test@test:~$ tar -zxf      image_filename.tar.gz
```

7) 在 Ubuntu PC 的图形界面双击 balenaEtcher- 1.5. 109-x64.AppImage 即可打开 balenaEtcher ，打开后的界面如下图所示

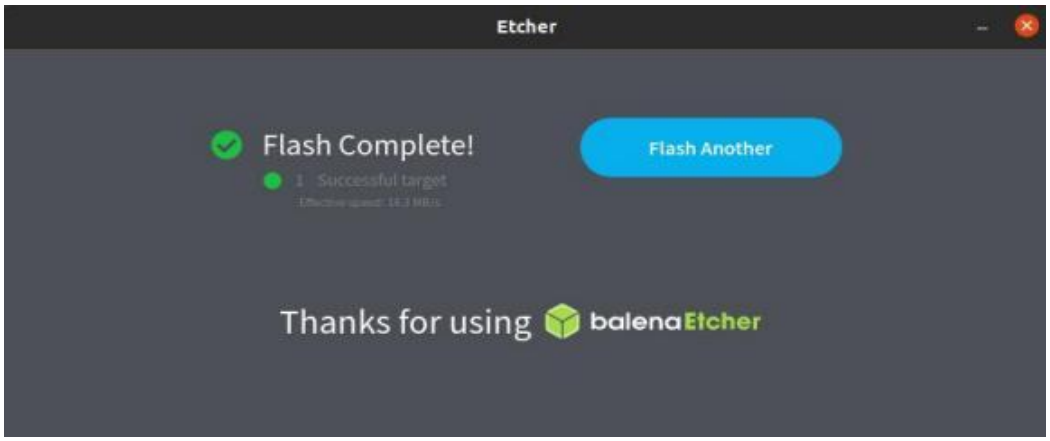
a. 首先选择镜像文件的路径

b. 然后选择 TF 卡的设备号

c. 最后点击 Flash 开始烧录进行

8) 烧录过程会提示写入的速度和剩余时间

9) 烧录完后会显示下面的界面，此时就可以把 TF 卡从电脑中拔出来插到开发板中启动了



2.5. 烧写 Android 固件到 TF 卡的方法

Android 镜像只能在 Windows 平台下使用 PhoenixCard 软件烧录到 TF 卡中，在 Linux 平台下无法烧录

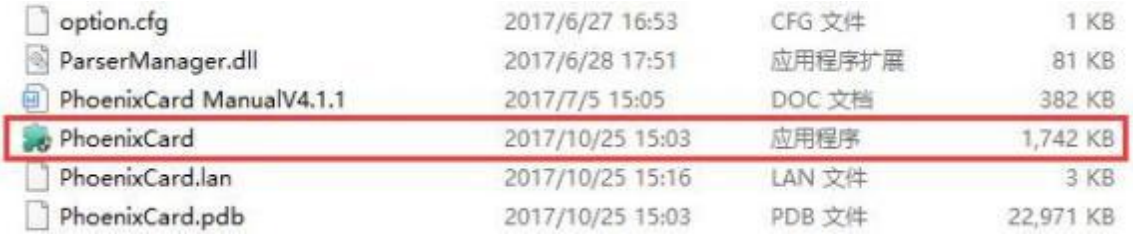
1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 class10 以上，建议使用闪迪等品牌的 TF 卡

2) 然后使用读卡器把 TF 卡插入电脑

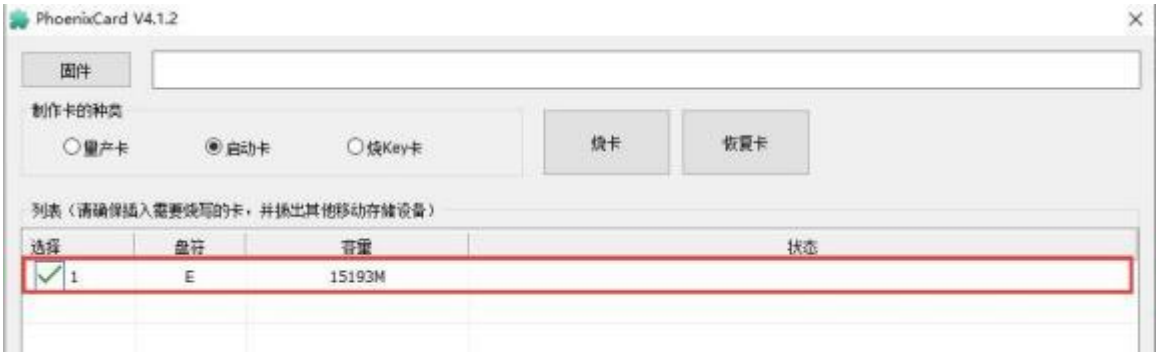
3) 从 WuKong Pi 的资料下载页面下载 Android 4.4 或者 Android 7.0 的固件和 PhoenixCard 烧写工具，请确保 PhonenixCrad 工具的版本为 PhoenixCard v4. 1.2

4) 使用解压软件解压下载的 Android 固件的压缩包，解压后的文件中，以“.img” 结尾的文件就是 Android 固件

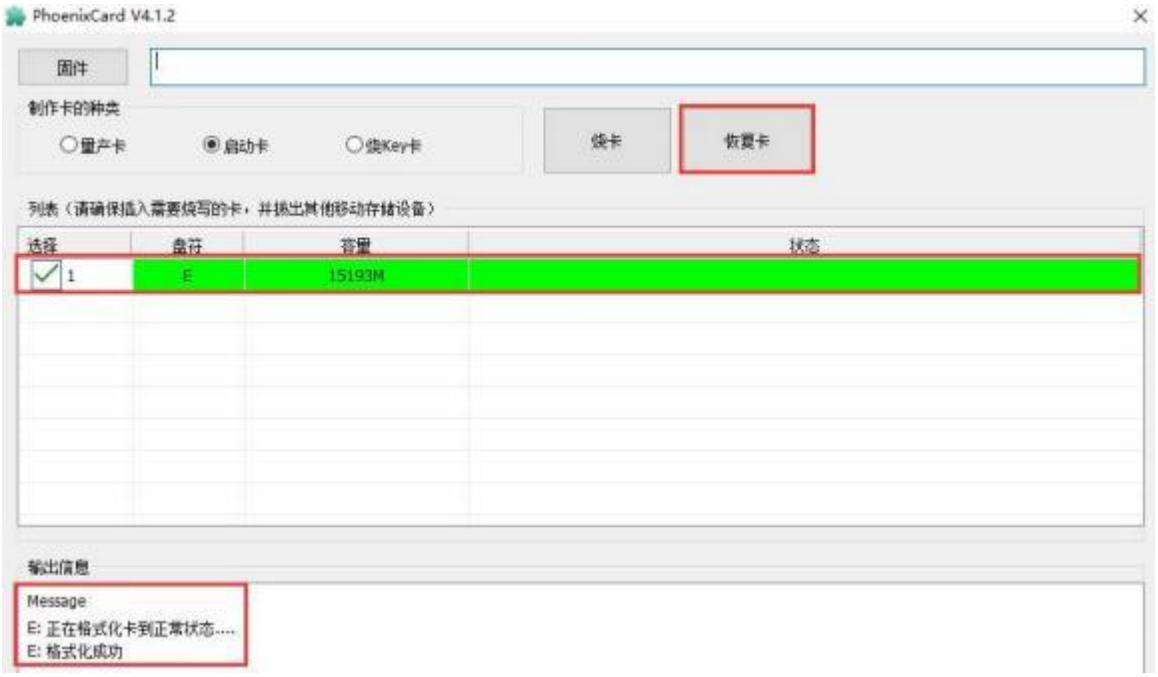
5) 使用解压软件解压 PhoenixCard v4. 1.2.rar ，此软件无需安装，在解压后的文件夹中找到 PhoenixCard 打开即可



6) 打开 PhoenixCard 后，如果 TF 卡识别正常，会在中间的列表中显示 TF 卡的盘符和容量，**请务必确认显示的盘符和你想烧录的TF 卡的盘符是一致的**，如果没有显 示可以尝试拔插下 TF 卡



7) 确认完盘符后，先格式化 TF 卡，点击PhoenixCard 中**恢复卡**按钮即可，也可以使用前面提到的 **SD Card Formatter** 进行 TF 卡的格式化

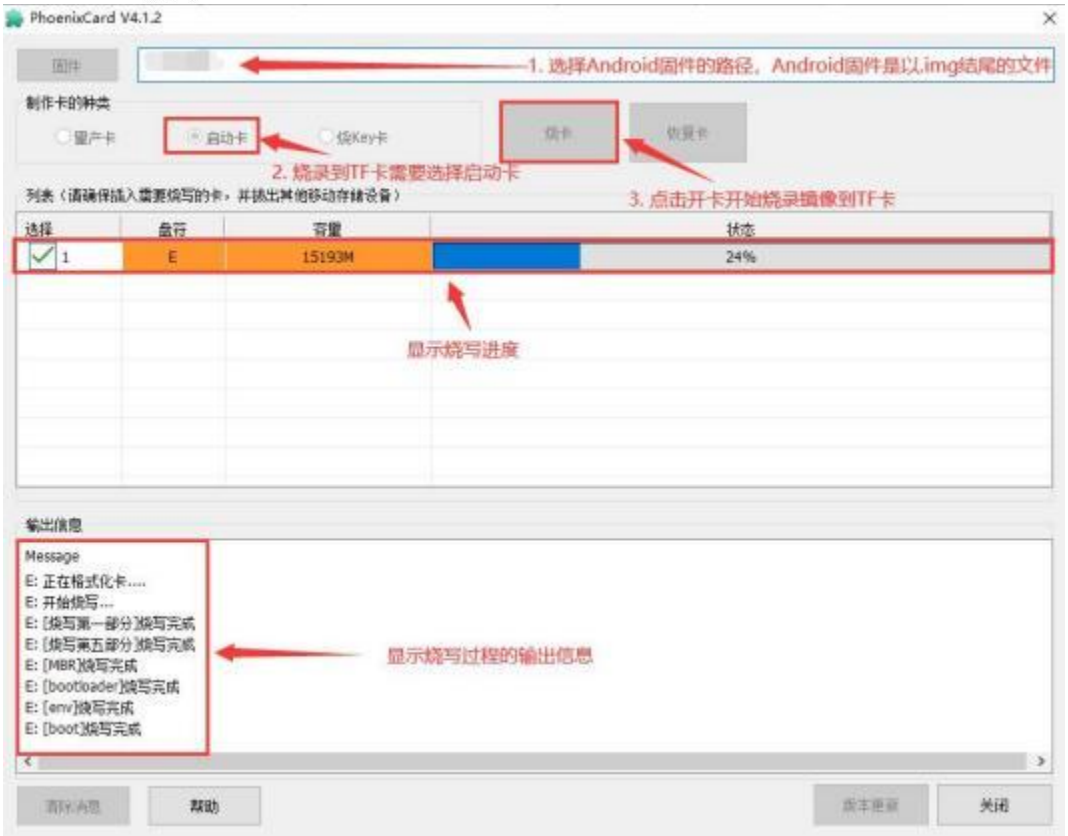


8）然后开始将Android 固件写入 TF 卡

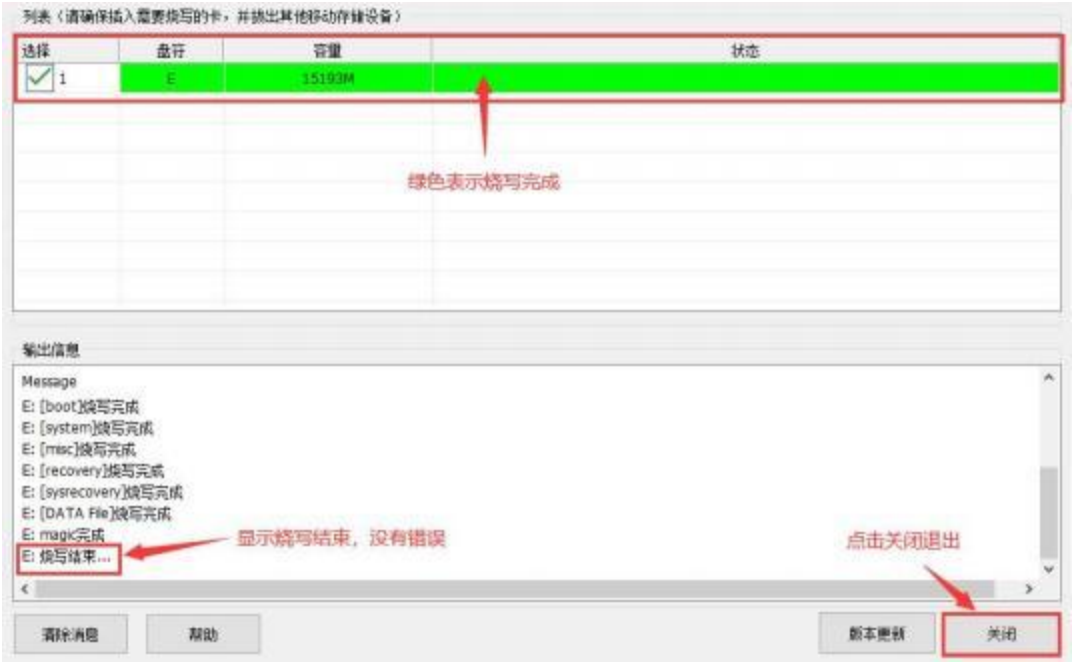
a. 首先在**固件**一栏中选择Android 固件的路径

b. 在**制作卡的种类**中选择**启动卡**

c. 然后点击**烧卡**按钮就会开始烧录



9) 烧录完后 PhoenixCard 的显示如下图所示，此时点击**关闭**按钮即可退出 PhoenixCard，然后就可以把 TF 卡从电脑中拔出来插到开发板中启动了



2.6. 启动悟空派开发板

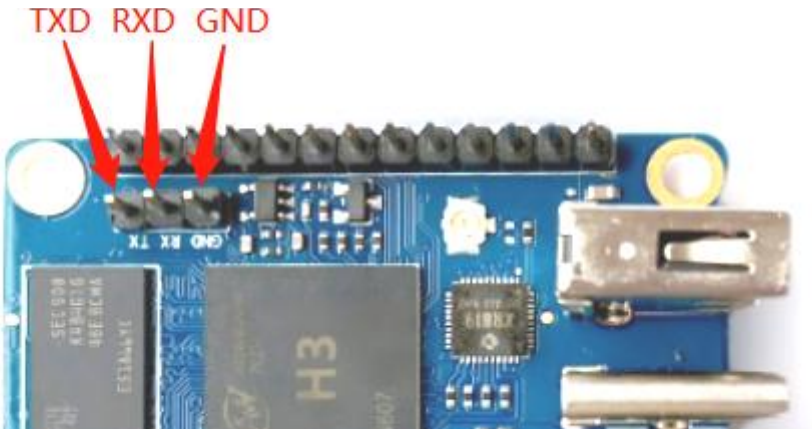
1) 将烧录好镜像的 TF 卡插入悟空派开发板的 TF 卡插槽
2) 如果购买了 13pin 的转接板，可以将 13pin 的转接板插到开发板的 13pin 中
3) 接上 USB 鼠标和键盘，用于控制悟空派开发板
4) 开发板有以太网口，可以插入网线用来上网
5) 连接一个 5V 和至少 2A（3A 的也可以）的电源适配器,Type-c接口
   a. **切记不要插入 12V 的电源适配器，如果插入了 12V 的电源适配器，会烧坏开发板**
   b. **系统上电启动过程中很多不稳定的现象基本都是供电有问题导致的，所以 一个靠谱的电源适配器很重要**
6) 然后打开电源适配器的开关，如果一切正常，此时系统就会正常启动了
7) 如果想通过调试串口查看系统的输出信息，请使用串口线将开发板连接到电脑，串口的连接方法请参看**调试串口的使用方法**一节

2.7. 调试串口的使用方法

2.7.1. 调试串口的连接说明

1) 首先需要准备一个 USB 转 TTL 模块，此模块在 WuKong Pi 的店铺中可以买到，如果有其他类似的 USB 转 TTL 模块也可以，然后将 USB 转 TTL 模块的 USB 一端 插入到电脑的 USB 接口中

2) 开发板的调试串口 GND 、TX 和 RX 引脚的对应关系如下图所示



3) USB 转 TTL 模块 GND、TXD 和 RXD 引脚需要通过杜邦线连接到开发板的调试串口上

a. USB 转 TTL 模块的 GND 接到开发板的 GND 上
b. USB 转 TTL 模块的 **RXD 接到开发板的 TXD 上**
c. USB 转 TTL 模块的 **TXD 接到开发板的 RXD 上**

4) USB 转 TTL 模块连接电脑和 WuKong Pi 开发板的示意图如下所示

USB TTL模块和WuKong Pi开发板的连接示意图

2.7.2. Ubuntu 平台调试串口的使用方法

1) 如果 USB 转 TTL 模块连接正常，在 Ubuntu PC 的/dev 下就可以看到对应的设备节点名，记住这个节点名，后面设置串口软件时会用到

```
test@test:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
```

2) linux 下可以使用的串口调试工具有很多，如 putty 、minicom 等，下面演示 putty 的使用方法

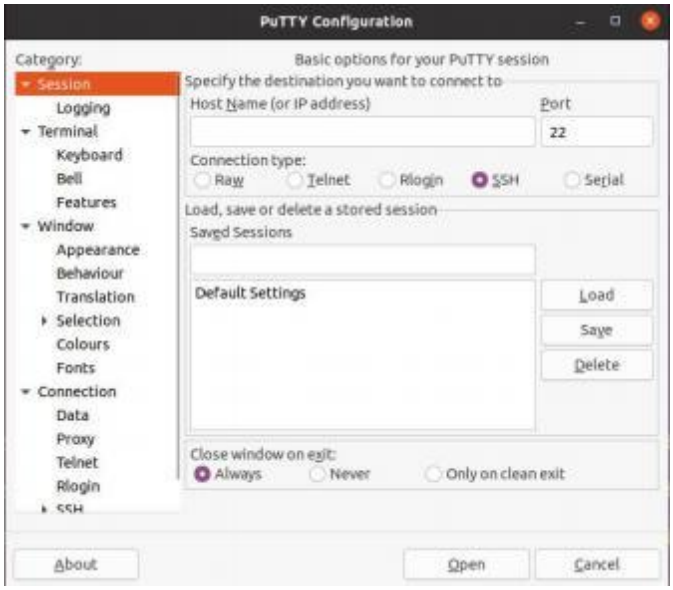
3) 首先在 Ubuntu PC 上安装 putty

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install putty
```

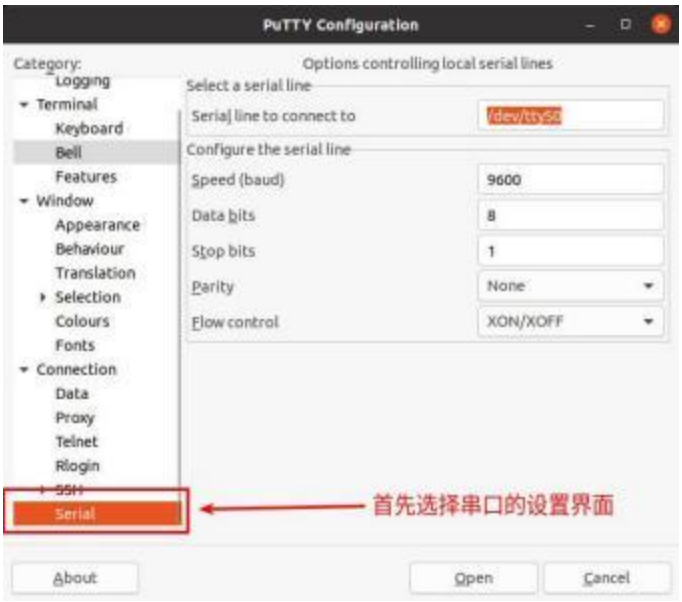
4) 然后运行 putty ，记得加 sudo 权限

```
test@test:~$ sudo putty
```

5) 执行 putty 命令后会弹出下面的界面



6) 首先选择串口的设置界面

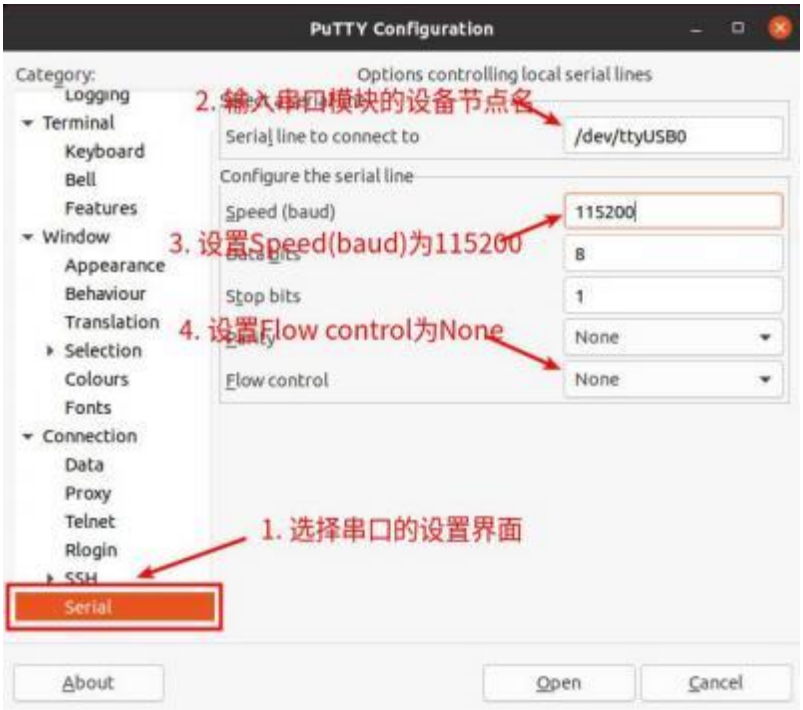


7) 然后设置串口的参数

a. 设置 Serial_line to_connect_to 为/dev/ttyUSB0 （修改为对应的节点名，一般为 /dev/ttyUSB0）

b. 设置 Speed(baud)为 115200

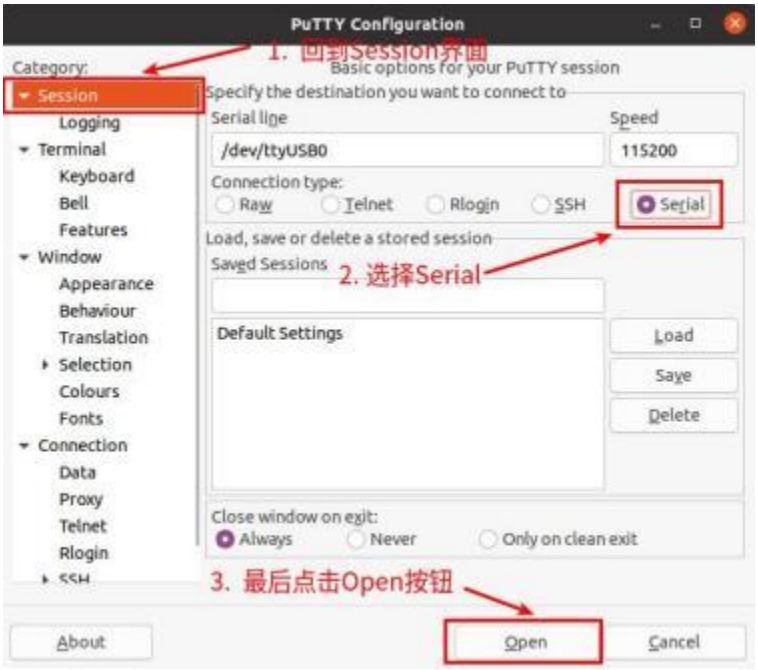
c. 设置 Flow control 为 None



8) 在串口的设置界面设置完后，在回到 Session 界面

a. 首先选择 Connection type 为 Serial

b. 然后点击 Open 按钮连接串口



9) 启动开发板后，就能从打开的串口终端中看到系统输出的 Log 信息了

2.7.3. Windows 平台调试串口的使用方法

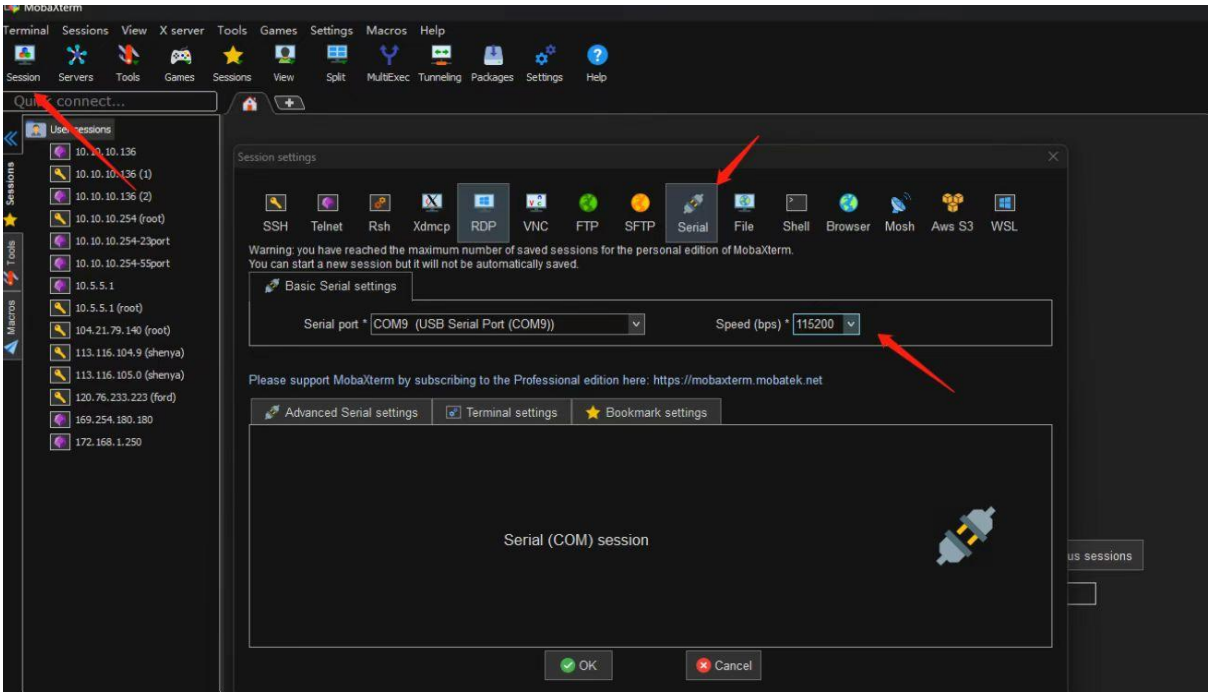
1) Windows 下可以使用的串口调试工具有很多，如 SecureCRT、MobaXterm 等，下 面演示 MobaXterm 的使用方法，这款软件是免费的，无需购买序列号即可使用

2) 下载 MobaXterm

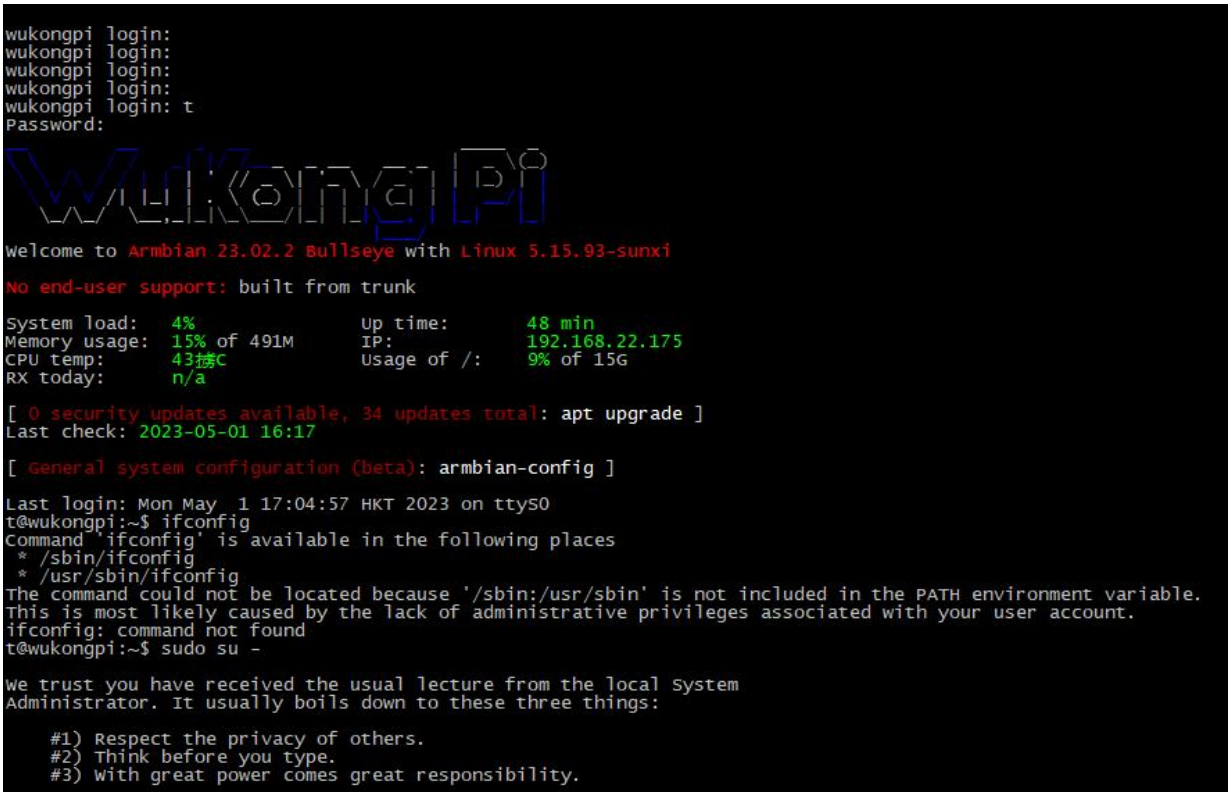
a. 下载 MobaXterm 网址如下

```
https://mobaxterm.mobatek.net/
```

2) MobaXterm的使用方法。选择Session->Serial->Speed(bps):115200.再选择一个串口。点OK即可



3) 启动开发板后，就能从打开的串口终端中看到系统输出的 Log 信息了



3. Linux 系统使用说明

3.1. 已支持的 linux 发行版类型和内核版本

发行版类型	内核版本	服务器版	桌面版
Ubuntu 20.04	linux5.4	支持	不支持
Ubuntu 18.04	linux5.4	支持	不支持
Debian 10	linux5.4	支持	不支持
Ubuntu 16.04	linux3.4	支持	支持

3.2. Linux5.4 内核镜像驱动适配情况

功能	状态
USB2.0 x 3	OK
TF 卡启动	OK
网卡	OK
红外	OK
WIFI	OK
耳机音频接口	OK
MIC 录音	OK
USB 摄像头	OK
LED 灯	OK
26pin GPIO	OK
I2C	OK
SPI	OK
UART	OK
温度传感器	OK
硬件看门狗	OK
TV-OUT	NO

3.3. Linux3.4 内核镜像驱动适配情况

功能	状态
USB2.0 x 3	OK
TF 卡启动	OK
网卡	OK
红外	OK
耳机音频接口	OK
USB 摄像头	OK
LED 灯	OK
26pin GPIO	OK
I2C	OK
SPI	OK
UART	OK
温度传感器	OK
硬件看门狗	OK
TV-OUT	OK

3.4. 登录账号和密码

首次登录需要设定root的用户名和密码，请牢记。

3.5. 板载 LED 灯显示控制说明

1) 开发板上有两个 LED 灯，一个绿灯，一个红灯，系统启动时 LED 灯默认显示情 况如下所示

	绿灯	红灯
u-boot 启动阶段	灭	亮
内核启动到进入系统	亮	灭或者闪烁
GPIO 口	PL10	PA17

2) 设置绿灯亮灭和闪烁的方法如下所示 （以linux3.4 系统为例）

a. 首先进入绿灯的设置目录

```
root@wukongpi:/sys/class/leds# cd wukongpi\:green\:pwr
```

b. 设置绿灯熄灭的命令如下

```
root@wukongpi:/sys/class/leds/wukongpi:green:pwr# echo 0 >brightness
```

c. 设置绿灯常亮的命令如下

```
root@wukongpi:/sys/class/leds/wukongpi:green:pwr# echo 1 >brightness
```

d. 设置绿灯闪烁的命令如下

```
root@wukongpi:/sys/class/leds/wukongpi:green:pwr# echo heartbeat > trigger
```

e. 设置绿灯停止闪烁的命令如下

```
root@wukongpi:/sys/class/leds/wukongpi:green:pwr# echo none > trigger
```

3) 设置红灯亮灭和闪烁的方法如下所示（以linux3.4 系统为例）

a. 首先进入红灯的设置目录

```
root@wukongpi:~# cd /sys/class/leds/wukongpi\:red\:status
```

b. 设置红灯熄灭的命令如下

```
root@wukongpi:/sys/class/leds/wukongpi:red:status# echo 0 > brightness
```

c. 设置红灯常亮的命令如下

```
root@wukongpi:/sys/class/leds/wukongpi:red:status# echo 1 > brightness
```

d. 设置红灯闪烁的命令如下

```
root@wukongpi:/sys/class/leds/wukongpi:red:status# echo heartbeat > trigger
```

e. 设置红灯停止闪烁的命令如下

```
root@wukongpi:/sys/class/leds/wukongpi:red:status# echo none > trigger
```

3.6. Linux5.4 系统第一次启动自动扩容 rootfs

1) 通过 TF 卡第一次启动 linux5.4 系统时会通过自动进行 rootfs 的扩容

2) 登录系统后可以通过df -h 命令来查看 rootfs 的大小，如果和 TF 卡的实际容量一 致，说明自动扩容运行正确

```
root@wukongpi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            430M     0  430M   0% /dev
tmpfs           100M  5.6M   95M   6% /run
/dev/mmcblk0p1   15G  915M   14G   7% /
tmpfs           500M     0  500M   0% /dev/shm
```

3) 需要注意的是，linux 系统只有一个ext4 格式的分区，没有使用单独的 BOOT 分区来存放内核镜像等文件，也就不存在 BOOT 分区扩容的问题

4) 另外如果不需要自动扩容 rootfs ，可以使用下面的方法来禁止

a. 首先将 linux 镜像烧录到 TF 卡中

b. 然后将 TF 卡插入 Ubuntu PC 中（Windows 不行），Ubuntu PC 一般会自动 挂载 TF 卡的分区，如果自动挂载正常，使用ls 命令可以看到下面的输出， TF 卡的分区名和下面命令所示名字不一定相同，请根据实际情况进行修改

```
test@test:~$ ls /media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/
bin   boot  dev   etc   home  lib   lost+found  media  mnt
opt   proc  root  run sbin  selinux  srv  sys  tmp  usr
var
```

c. 然后在 Ubuntu PC 中将当前用户切换成 root 用户

```
test@test:~$ sudo -i
[sudo] test 的密码：
root@test:~#
```

d. 然后进入 TF 卡中的 linux 系统的 root 目录下新建一个名为 **.no_rootfs_resize** 的文件

```
root@test:~# cd /media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db# cd root
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/root# touch .no_rootfs_resize
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/root# ls .no_rootfs*
.no_rootfs_resize
```

e. 然后就可以卸载 TF 卡，再拔出TF 插到开发板启动，linux 系统启动时，当 检测到/root 目录下有**.no_rootfs_resize** 这个文件就不会再自动扩容 rootfs 了

f. 禁止 rootfs 自动扩容后可以看到 TF 卡可用容量只有 200M 左右

```
root@wukongpi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            927M     0  927M   0% /dev
tmpfs           200M  5.6M  194M   3% /run
/dev/mmcblk0p1  1.5G  1.3G  196M  87% /
tmpfs           997M     0  997M   0% /dev/shm
```

3.8. 修改 linux 日志级别（loglevel）的方法

1) linux 系统的 loglevel 默认设置为 1 ，当使用串口查看启动信息时，会屏蔽部分内 核的输出 log

2) 当 linux 系统启动出现问题时，可以使用下面的方法来修改 loglevel 的值，从而 打印更多的 log 信息到串口显示，方便调试。如果 linux 系统启动失败，无法进入系 统，可以把 TF 卡通过读卡器插入 Ubuntu PC 中，然后在 Ubuntu PC 中挂载 TF 卡后 直接修改 TF 卡中的 linux 系统的配置，修改完后，再把 TF 卡插入开发板中启动

```
root@wukongpi:~# sed -i "s/verbosity=1/verbosity=7/" /boot/armbianEnv.txt
root@wukongpi:~# sed -i "s/console=both/console=serial/"
/boot/wukongpiEnv.txt
```

3) 上面的命令其实都是设置**/boot/armbianEnv.txt** 中的变量，设置完后可以打开 **/boot/armbianEnv.txt** 检查下

```
root@wukongpi:~# cat /boot/armbianEnv.txt
verbosity=7
bootlogo=false
console=serial
disp_mode=1920x1080p60
overlay_prefix=sun8i-h3
overlays=usbhost2 usbhost3
rootdev=UUID=2a49e4b2-7afa-48bf-bcf1-a02174fc1ad6
rootfstype=ext4
usbstoragequirks=0x2537:0x1066:u,0x2537:0x1068:u
```

4) 然后重启开发板，内核的输出信息就都会打印到串口输出了

3.9. 以太网口测试

1) 首先将网线插入开发板的以太网口，并确保网络是畅通
2) 系统启动后会通过 DHCP 自动给以太网卡分配 IP 地址
3) 查看 IP 地址的命令如下

```
root@wukongpi:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.100.11  netmask 255.255.255.0  broadcast 192.168.100.255
inet6 fe80::668a:641:15f3:dab2  prefixlen 64  scopeid 0x20<link>
ether 02:81:aa:6e:dc:8c  txqueuelen 1000  (Ethernet)
RX packets 91  bytes 11232 (10.9 KiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 41  bytes 3896 (3.8 KiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
device interrupt 47
```

4) 测试网络连通性的命令如下

```
root@wukongpi:~# ping www.baidu.com -I eth0
PING www.a.shifen.com (14.119.104.189) from 192.168.100.11 eth0: 56(84) bytes of data.
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=1 ttl=55 time=8.47 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=2 ttl=55 time=7.62 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=3 ttl=55 time=7.68 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=4 ttl=55 time=7.50 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=5 ttl=55 time=8.03 ms
^C
--- www.a.shifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 7.499/7.861/8.472/0.353 ms
```

3.10. SSH 远程登录开发板

Linux 系统默认都开启了 ssh 远程登录，并且允许 root 用户登录系统。ssh 登录前首先需要确保以太网或者 wifi 网络已连接，然后使用ifconfig 命令或者通过查看 路由器的方式获取开发板的IP 地址

3.10.1. Ubuntu 下 SSH 远程登录开发板

1) 首先获取开发板的IP 地址

2) 然后就可以通过 ssh 命令远程登录 linux 系统

```
test@test:~$ ssh root@192.168.100.11      //需要替换为开发板的IP 地址
root@192.168.100.11's password:           //在这里输入密码，我设置默认密码为root
```

3) 这样就可以成功登录系统了。

4) 如果 ssh 登录时提示下面的错误

```
test@test:~$ ssh root@192.168.100.11
Connection reset by 192. 168. 100. 149 port 22
lost connection
```

可以在开发板上输入下面的命令再尝试是否能连接

```
root@wukongpi:~# rm /etc/ssh/ssh_host_*
root@wukongpi:~# dpkg-reconfigure openssh-server
```

3.10.2. Windows 下 SSH 远程登录开发板

1) 首先获取开发板的IP地址

2) 在 windows 下可以使用MobaXterm远程登录开发板，首先新建一个 ssh 会话

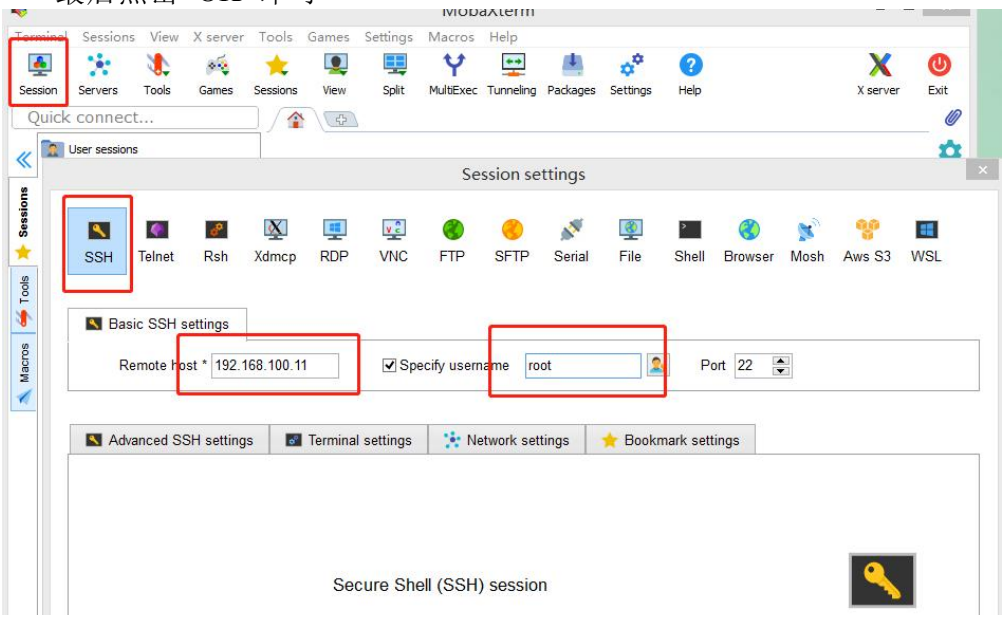
a. 打开 **Session**

b. 然后在 **Session Setting** 中选择 **SSH**

c. 然后在 **Remote host** 中输入开发板的 IP 地址

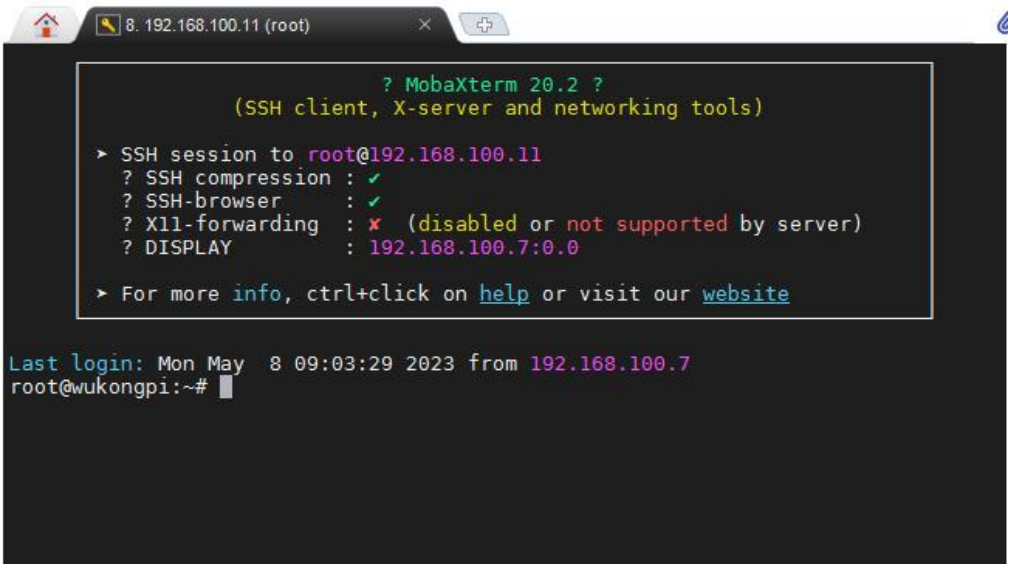
d. 然后 **Specify username** 中输入 linux 系统的用户名 **root** 或 **wukongpi**

e. 最后点击 OK 即可



3) 然后会提示输入密码，默认 root 和 wukongpi 用户的密码都为root

4) 成功登录系统后的显示如下图所示



3.11. WIFI 连接测试

请不要通过修改/etc/network/interfaces 配置文件的方式来连接 WIFI ，通过这种方式连接 WIFI 网络使用会有问题

3.11.1. 服务器版镜像通过命令连接 WIFI

当开发板没有连接以太网，只连接了串口时，推荐使用此小节演示的命令来连接 WIFI 网络。因为 nmtui 在某些串口软件（如 minicom）中只能显示字符，无法正 常显示图形界面。当然，如果开发板连接了以太网，也可以使用此小节演示的命令 来连接 WIFI 网络的

1) 先登录 linux 系统，有下面三种方式

a. 如果开发板连接了网线，可以通过ssh 远程登录 linux 系统

b. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统

2) 首先使用 nmcli dev wifi 命令扫描周围的 WIFI 热点

```
root@wukongpi:~# nmcli dev wifi
```

3) 然后使用 **nmcli** 命令连接扫描到的 WIFI 热点，其中：

a. **wifi_name** 需要换成想连接的 WIFI 热点的名字

b. **wifi_passwd** 需要换成想连接的 WIFI 热点的密码

root@wukongpi:~# nmcli dev wifi connect **wifi_name** password **wifi_passwd**
Error: No network with SSID 'wifi_name' found.

例如：

root@wukongpi:~# nmcli dev wifi connect Hi-Link-GC password 12345678
Device 'wlan0' successfully activated with '559ee6d8-90d3-4f12-905e-a1cb2f9afd06'.

4) 通过 ifconfig 命令可以查看 wifi 的 IP 地址

root@wukongpi:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.100.12 netmask 255.255.255.0 broadcast 192.168.100.255
inet6 fe80::d2a8:27ee:1701:a174 prefixlen 64 scopeid 0x20<link>
ether 12:81:aa:6e:dc:8c txqueuelen 1000 (Ethernet)
RX packets 1908 bytes 761938 (744.0 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 22 bytes 3170 (3.0 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

5) 使用 ping 命令可以测试 wifi 网络的连通性

```
root@wukongpi:~# ping www.baidu.com -I wlan0
PING www.a.shifen.com (14.119.104.189) from 192.168.100.12 wlan0: 56(84) bytes of data.
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=1 ttl=55 time=11.9 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=2 ttl=55 time=14.2 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=3 ttl=55 time=12.2 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=4 ttl=55 time=23.2 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=5 ttl=55 time=16.0 ms
^C
--- www.a.shifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4043ms
rtt min/avg/max/mdev = 11.910/15.498/23.209/4.133 ms
```

3.11.2. 服务器版镜像通过图形化方式连接 WIFI

1) 先登录 linux 系统，有下面三种方式

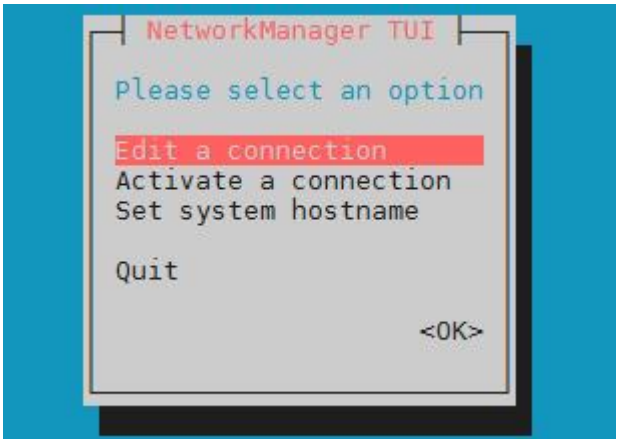
   a. 如果开发板连接了网线，可以通过ssh 远程登录 linux 系统

   b. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统（串口软 件请使用 MobaXterm ，使用 minicom 无法显示图形界面）

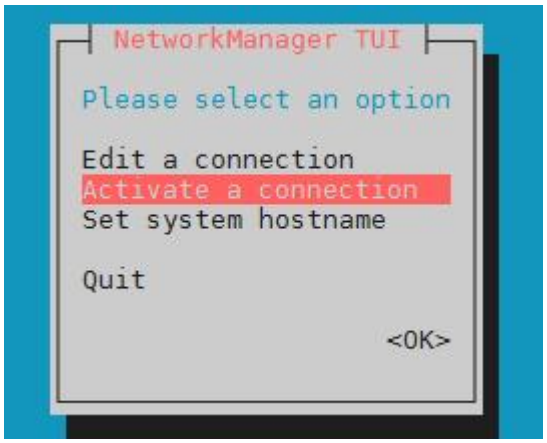
2) 然后在命令行中输入 nmtui 命令打开 wifi 连接的界面

```
root@wukongpi:~# nmtui
```

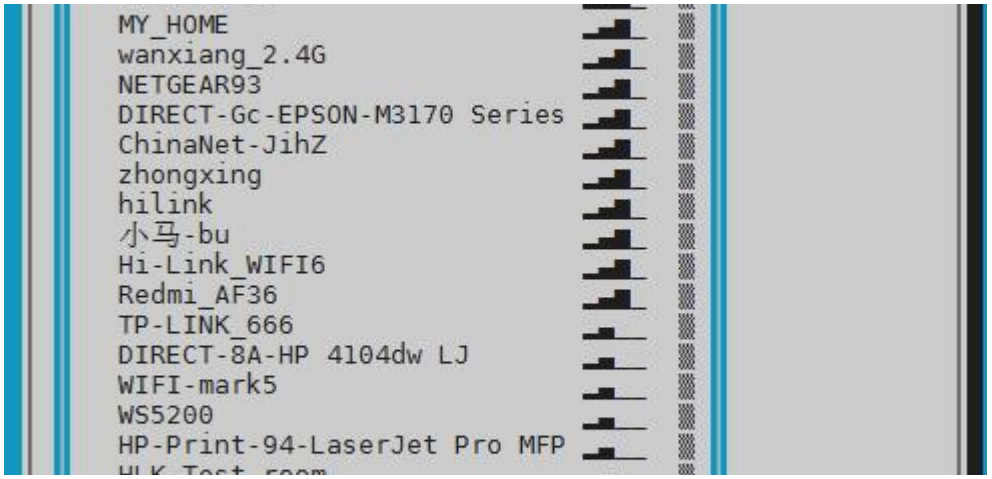
3) 输入 nmtui 命令打开的界面如下所示



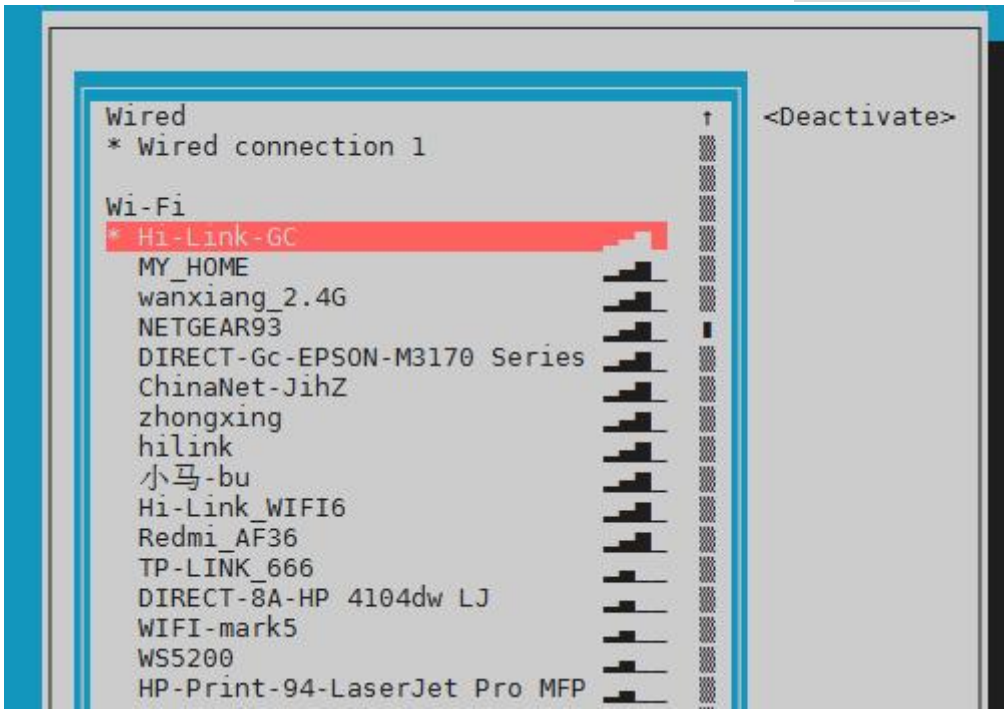
4) 选择 **Activate a connect** 后回车



5) 然后就能看到所有搜索到的 WIFI 热点



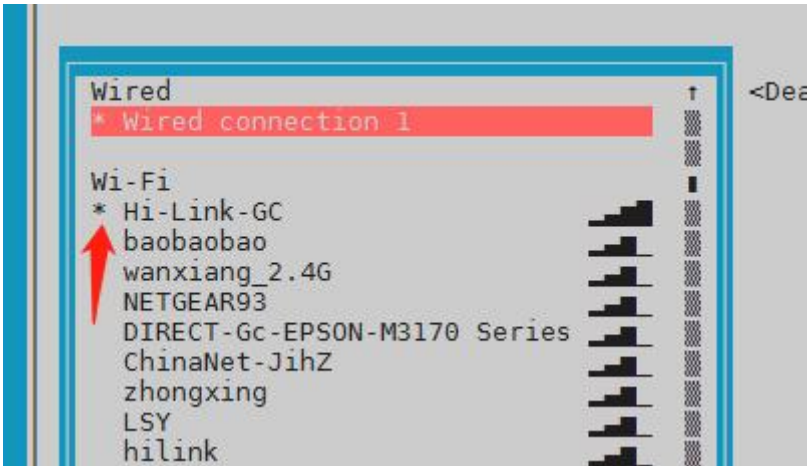
6) 选择想要连接的 WIFI 热点后再使用 Tab 键将光标定位到 **Activate** 后回车



7) 然后会弹出输入密码的对话框，在Pssword 内输入对应的密码然后回车就会开始

连接 WIFI

8) WIFI 连接成功后会在已连接的 WIFI 名称前显示一个“*”



9) 通过 ifconfig 命令可以查看 wifi 的 IP 地址

```
root@wukongpi:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.12  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::d2a8:27ee:1701:a174  prefixlen 64  scopeid 0x20<link>
        ether 12:81:aa:6e:dc:8c  txqueuelen 1000  (Ethernet)
        RX packets 1908  bytes 761938 (744.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22  bytes 3170 (3.0 KiB)
                TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3.12. USB 接口测试

1) 先格式化 U 盘，然后在 U 盘中放入一些文件

2) 再将 U 盘插入悟空派H3的USB 接口中

4) 悟空派H3 Zero 接上转接板后总共有 3 个 USB 接口可以使用，linux5.4 系统使用 转接板上的两个 USB 接口前需要确保/boot/armbianEnv.txt 中的 overlays 配置了 usbhost2 和 usbhost3 才能正常使用，linux5.4 系统默认都打开了 usbhost2 和 usbhost3。 如果没有配置，转接板上的两个 USB 接口是无法正常使用的，配置完后需要重启才 能生效。另外 linux3.4 系统无需检查，默认 3 个 USB 接口都是打开的

```
root@wukongpi:~# cat /boot/armbianEnv.txt
overlays=usbhost2 usbhost3
```

5) 执行下面的命令如果能看到 sdX 的输出说明U 盘识别成功

```
root@wukongpi:~# cat /proc/partitions | grep "sd*"
major minor  #blocks  name
   8        0   30044160 sda
   8        1   30043119 sda1
```

6) 使用 mount 命令可以将 U 盘挂载到/mnt 中，然后就能查看 U 盘中的文件了

```
root@wukongpi:~# mount /dev/sda1 /mnt/
root@wukongpi:~# ls /mnt/
test.txt
```

7) 挂载完后通过df 命令能查看 U 盘的容量使用情况和挂载点

```
root@wukongpi:~# df -h | grep "sd"
/dev/sda1                29G  208K   29G   1% /mnt
```

3.13. USB 以太网卡测试

1) 目前**测试过**能用的 USB 以太网卡如下所示，其中RTL8153 USB 千兆网卡插入开发板的USB 2.0 Host 接口中测试可以正常使用，但是速率是达不到千兆的，这点请注意

序号	型号
1	RTL8152B USB 百兆网卡
2	RTL8153 USB 千兆网卡

2) 首先将 USB 网卡插入开发板的 USB 接口中，然后在 USB 网卡中插入网线，确 保网线能正常上网，如果通过 dmesg 命令可以看到下面的 log 信息，说明 USB 网卡识别正常

```
root@wukongpi:~# dmesg | tail
[  121.985016] usb 3-1: USB disconnect, device number 2
[  126.873772] sunxi-ehci 5311000.ehci3-controller: ehci_irq: highspeed device connect [  127.094054] usb 3-1: new high-speed USB device number 3 using sunxi-ehci [  127.357472] usb 3-1: reset high-speed USB device number 3 using sunxi-ehci [  127.557960] r8152 3-1:1.0 eth1: v1.08.9
[  127.602642] r8152 3-1:1.0 enx00e04c362017: renamed from eth1
[  127.731874] IPv6: ADDRCONF(NETDEV_UP): enx00e04c362017: link is not ready
```

```
[       127.763031] IPv6: ADDRCONF(NETDEV_UP): enx00e04c362017: link is not
ready [       129.892465] r8152 3- 1:1.0 enx00e04c362017: carrier on
[            129.892583]   IPv6:   ADDRCONF(NETDEV_CHANGE):
enx00e04c362017:   link becomes  ready
```

3) 然后通过 ifconfig 命令可以看到 USB 网卡的设备节点，以及自动分配的 IP 地址

4) 测试网络连通性的命令参照之前的。

3.14. USB 摄像头测试

1) 先将 13pin 转接板插入到开发板的 13pin 功能接口中，再启动 Linux 系统

2) 然后将 USB 摄像头插入到 WuKong Pi 的 USB 接口中

3) 悟空派H3 Zero 接上转接板后总共有 3 个 USB 接口可以使用，linux5.4 系统使用转接板上的两个 USB 接口前需要确保/boot/armbianEnv.txt 中的 overlays 配置了 usbhost2 和 usbhost3 才能正常使用，linux5.4 系统默认都打开了 usbhost2 和 usbhost3。 如果没有配置，转接板上的两个 USB 接口是无法正常使用的，配置完后需要重启才 能生效。另外 linux3.4 系统无需检查，默认 3 个 USB 接口都是打开的

```
root@wukongpi:~# cat /boot/armbianEnv.txt
overlays=usbhost2 usbhost3
```

4) 然后使用lsmod 查看系统是否自动加载了 uvcvideo 内核模块

```
root@wukongpi:~# lsmod
Module                  Size  Used by
uvcvideo              106496  0
```

5) 然后通过 v4l2-ctl （**注意 v4l2 中的 l 是小写字母 l ，不是数字 1**）命令查看下 USB 摄像头的设备节点，从下面的输出可知 USB 摄像头对应的设备节点为**/dev/video1**，如果看不到 usb 相关的 video 节点，说明 USB 摄像头无法识别

```
root@wukongpi:~# apt update
root@wukongpi:~# apt install v4l-utils
root@wukongpi:~# v4l2-ctl       --list-devices
cedrus (platform:cedrus):
    /dev/video0

USB 2.0 Camera: HD USB Camera (usb-1c1d000.usb-1):
    /dev/video1
    /dev/video2
```

6) 安装 fswebcam

```
root@wukongpi:~# apt update
root@wukongpi:~# apt-get install fswebcam
```

7) 安装完 fswebcam 后可以使用下面的命令来拍照

a. -d 选项用于指定 USB 摄像头的设备节点
b. --no-banner 用于去除照片的水印
c. -r 选项用于指定照片的分辨率
d. -S 选项用设置于跳过前面的帧数

```
root@wukongpi:~# fswebcam -d /dev/video2 --no-banner -r 1280x720 -S 5 ./image.jpg
```

8) 在服务器版的 Linux 系统中，拍完照后可以使用 scp 命令将拍好的图片传到 Ubuntu PC 上镜像观看

```
root@wukongpi:~# scp image.jpg test@192.168.1.55:/home/test //需要修改为对应的路径
```

4. Linux SDK 使用说明

Linux SDK 的编译都是在安装有 **Ubuntu 18.04 的 PC 或者虚拟机(VirtualBox 或VMware)**上进行的，请不要使用其他版本的Ubuntu 系统或者在WSL上编译Linux
SDK

4.1. 获取 linux sdk 的源码

4.1.1. 从 github 下载 wukongpi-build

1) 首先下载 wukongpi-build 的代码，wukongpi-build 的代码是基于 armbian build 编译
系统修改而来的， 目前 H3 系列开发板已经支持 legacy 分支和 current 分支

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install git
test@test:~$ git clone https://github.com/Timfu2019/wukongpi_build.git
```

通过 git clone 命令下载 wukongpi-build 的代码是不需要输入 github 账号的用户名 和密码的 （下载本手册中的其他代码也是一样的），如果如输入 git clone 命令后 Ubuntu PC 提示需要输入 github 账号的用户名和密码，一般都是 git clone 后面的 wukongpi-build 仓库的地址输入错误了，首先请仔细检查命令拼写是否有错误，而不是以为我们这里忘了提供 github 账号的用户名和密码

2) wukongpi-build 下载完后会包含下面的文件和文件夹

a. **build.sh**: 编译启动脚本
b. **external**: 包含编译镜像需要用的配置文件、特定的脚本以及部分程序的源 码等
c. **LICENSE**: GPL 2 许可证文件,GPL-2.0 license
d. **README.md**: wukongpi-build 说明文件
e. **scripts**: 编译 linux 镜像的通用脚本

```
test@test:~/wukongpi-build$ ls
build.sh       external       LICENSE       README.md  scripts
```

4.1.2. 下载交叉编译工具链

1) wukongpi-build 第一次运行的时候会自动下载交叉编译工具链放在**toolchains** 文件 夹中，每次运行wukongpi-build 的 build.sh 脚本后，都会检查 **toolchains** 中的交叉编 译工具链是否都存在，如果不存在则会重新开始下载，如果存在则直接使用，不会重复下载

2) 交叉编译工具链在中国境内的镜像网址为清华大学的开源软件镜像站

```
https://mirrors.tuna.tsinghua.edu.cn/armbian-releases/_toolchain/
```

3) **toolchains** 下载完后会包含多个版本的交叉编译工具链

```
test@test:~/wukongpi-build$ ls toolchains/
gcc-arm-9.2-2019. 12-x86_64-aarch64-none-linux-gnu
gcc-arm-9.2-2019. 12-x86_64-arm-none-linux-gnueabihf
gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi
gcc-linaro-5.5.0-2017. 10-x86_64_arm-linux-gnueabihf
gcc-linaro-7.4. 1-2019.02-x86_64_aarch64-linux-gnu
gcc-linaro-7.4. 1-2019.02-x86_64_arm-linux-gnueabi
gcc-linaro-aarch64-none-elf-4.8-2013. 11_linux
gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux
gcc-linaro-arm-none-eabi-4.8-2014.04_linux
```

4) 编译 H3 linux 内核源码使用的交叉编译工具链为

a. linux3.4

```
gcc-linaro-5.5.0-2017. 10-x86_64_arm-linux-gnueabihf
```

b. linux5.4

```
gcc-arm-9.2-2019. 12-x86_64-arm-none-linux-gnueabihf
```

5) 编译 H3 u-boot 源码使用的交叉编译工具链为

a. u-boot 2018.05

```
gcc-arm-9.2-2019. 12-x86_64-arm-none-linux-gnueabihf
```

b. u-boot 2020.04

```
gcc-arm-9.2-2019. 12-x86_64-arm-none-linux-gnueabihf
```

4.1.3. wukongpi-build 完整目录结构说明

1) wukongpi-build 仓库下载完后并不包含 linux 内核、u-boot 的源码以及交叉编译工 具链，linux 内核和 u-boot 的源码存放在独立的 git 仓库中（**请不要单独下载使用内 核和 u-boot 的源码进行编译操作，除非你知道怎么用**）

4.1.4. 从百度云盘下载 wukongpi-build

1) 如果从 github 下载 wukongpi-build 的速度很慢，还可以从百度云盘下载 wukongpi-build 的压缩包，百度云盘的下载链接为

```
https://pan.baidu.com/s/1scMeadmTEeOpsAw2o_tCJg?pwd=abcd
```

2) 百度云盘的 wukongpi-build 文件夹下有两个文件

a. **wukongpi-build.tar.gz** 为 wukongpi-build 源码的压缩包

b. **wukongpi-build.tar.gz.md5sum** 为 wukongpi-build 源码的压缩包的 MD5 校验 和文件

c. 下载完后，请首先检查下 wukongpi-build.tar.gz 压缩包的 MD5 校验和是否正确，这样可以防止下载的压缩包有问题，如果不正确，请重新下载，检查校 验和是否正确的命令为

```
test@test:~$ md5sum -c wukongpi-build.tar.gz.md5sum
wukongpi-build.tar.gz:  成功
```

3) 然后就可以使用tar-zxf命令解压 wukongpi-build.tar.gz

```
test@test:~$ tar -zxf wukongpi-build.tar.gz
test@test:~$ cd wukongpi-build/
test@test:~/wukongpi-build$ ls
build.sh        external        kernel        LICENSE        README.md
scripts         toolchains
u-boot          userpatches
```

4) 使用 wukongpi-build 编译系统前，请先将 wukongpi-build 和 github 服务器进行同步， 确保代码为最新的状态

```
test@test:~/wukongpi-build$ git pull
```

如果 wukongpi-build 和 github 服务器同步过程有问题，可以尝试下面的方法

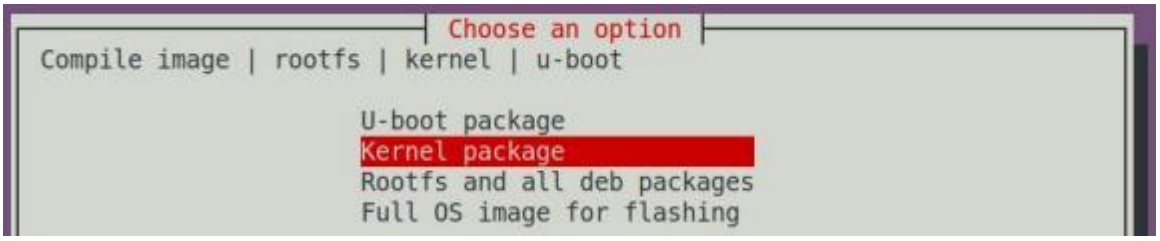
```
test@test:~/wukongpi-build$ sudo rm -rf external scripts
test@test:~/wukongpi-build$ git checkout .
test@test:~/wukongpi-build$ git pull
```

4.3. 编译 linux 内核

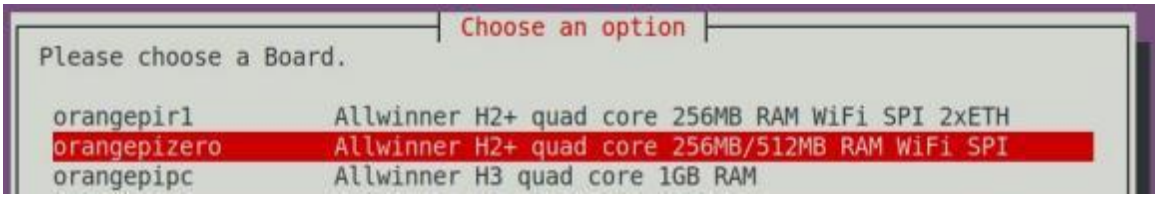
1) 运行 build.sh 脚本，记得加 sudo 权限

```
test@test:~/wukongpi-build$ sudo ./build.sh
```

2) 选择 **Kernel package** ，然后回车



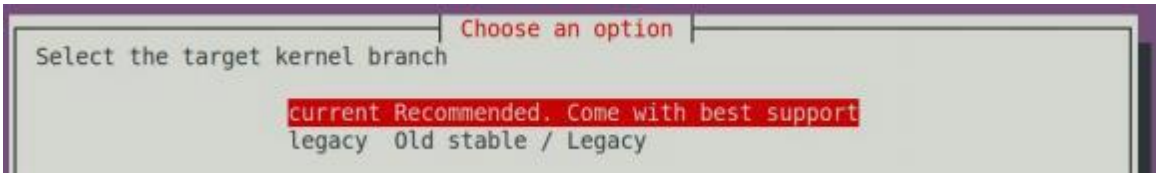
3) 接着选择开发板的型号



4) 然后选择分支

a. current 会编译 linux5.4

b. legacy 会编译 linux3.4



5) 然后会弹出通过 **make_menuconfig** 打开的内核配置的界面，此时可以直接修改内核的配置，如果不需要修改内核配置，直接退出即可，退出后会开始编译内核源码

a. 如果不需要修改内核的配置选项，在运行 build.sh 脚本时，传入 **KERNEL_CONFIGURE=no** 就可临时屏蔽弹出内核的配置界面了

```
test@test:~/wukongpi-build$ sudo ./build.sh KERNEL_CONFIGURE=no
```

b. 也可以设置 wukongpi-build/userpatches/config-default.conf 配置文件中的 **KERNEL_CONFIGURE=no** ，这样可以永久禁用这个功能

c. 编译内核的时候如果提示下面的错误，这是由于 Ubuntu PC 的终端界面太 小，导致 make menuconfig 的界面无法显示，请把 Ubuntu PC 的终端调到最大，然 后重新运行build.sh 脚本

```
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
scripts/kconfig/Makefile:28: recipe for target 'menuconfig' failed
make[1]: *** [menuconfig] Error 1
Makefile:560: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
[ error ] ERROR in function compile_kernel [ compilation.sh:376 ]
[ error ] Error kernel menuconfig failed
[ o.k. ] Process terminated
```

6) 编译内核源码时会提示下面的信息（以 current 分支为例）

a. 内核源码的版本

[o.k.] Compiling legacy kernel [**5.4.65**]

b. 编译内核源码使用的交叉编译工具链的版本

[o.k.] Compiler version [**aarch64-none-linux-gnu-gcc 9.2.1**]

i. 内核默认使用的配置文件以及它存放的路径

[o.k.] Using kernel config file [**config/kernel/linux-sunxi-current.config**]

j. 内核最终使用的配置文件.config （通过 make menuconfig 对默认的内核配置 文件进行了修改）会复制到 **output/config** 中，如果没有对内核配置进行修 改，最终的配置文件和默认的配置文件是一致的

[o.k.] Exporting new kernel config [**output/config/linux-sunxi-current.config**]

k. 编译生成的内核相关的 deb 包的路径

[o.k.] Target directory [**output/debs/**]

l. 编译生成的包含内核镜像和内核模块的deb 包的包名

[o.k.] File name [**linux-image-current-sunxi_2.1.0_armhf.deb**]

m. 编译使用的时间

[o.k.] Runtime [**4 min**]

n. 最后会显示重复编译上一次选择的内核的编译命令，使用下面的命令无需通 过图形界面选择，可以直接开始编译内核源码

[o.k.] Repeat Build Options [**sudo ./build.sh BOARD=wukongpizero BRANCH=current BUILD_OPT=kernel KERNEL_CONFIGURE=yes**]

10) 查看编译生成的内核镜像相关的deb 包

a. **linux-dtb-current-sunxi_2.1.0_armhf.deb** 包含内核使用的 dtb 文件

b. **linux-headers-current-sunxi_2.1.0_armhf.deb** 包含内核使用的头文件

c. **linux-image-current-sunxi_2.1.0_armhf.deb** 包含内核镜像和内核模块

```
test@test:~/wukongpi-build$ ls output/debs/linux-*
output/debs/linux-dtb-current-sunxi_2. 1.0_armhf.deb
output/debs/linux-image-current-sunxi_2. 1.0_armhf.deb
output/debs/linux-headers-current-sunxi_2. 1.0_armhf.deb
```

11) 生成的 linux-image 的 deb 包包含的文件如下所示

a. 使用下面的命令可以解压 deb 包

```
test@test:~/wukongpi-build$ cd output/debs
test@test:~/wukongpi_build/output/debs$ mkdir test
test@test:~/wukongpi_build/output/debs$ cp \
linux-image-current-sunxi_2.1.0_armhf.deb test/
test@test:~/wukongpi_build/output/debs$ cd test
test@test:~/wukongpi_build/output/debs/test$ dpkg -x \
linux-image-current-sunxi_2.1.0_armhf.deb .
test@test:~/wukongpi_build/output/debs/test$ ls
boot    etc    lib    linux-image-current-sunxi_2. 1.0_armhf.deb    usr
```

b. 解压后的文件如下所示

```
test@test:~/wukongpi_build/output/debs/test$ tree -L 2
.
├── boot
│   ├── config-5.4.65-sunxi                    //编译内核源码使用的配置文件
│   ├── System.map-5.4.65-sunxi
│   └── vmlinuz-5.4.65-sunxi                   //编译生成的内核镜像文件
├── etc
│   └── kernel
├── lib
│   └── modules                                          //编译生成的内核模块
├── linux-image-current-sunxi_2. 1.0_armhf.deb
└── usr
    ├── lib
    └── share

8 directories, 4 files
```

12) 生成的 linux-dtb 的 deb 包包含的文件如下所示

a. 使用下面的命令可以解压 deb 包

```
test@test:~/wukongpi-build$ cd output/debs
test@test:~/wukongpi_build/output/debs$ mkdir test
test@test:~/wukongpi_build/output/debs$ cp \
linux-dtb-current-sunxi_2.1.0_armhf.deb test/
test@test:~/wukongpi_build/output/debs$ cd test
test@test:~/wukongpi_build/output/debs/test$ dpkg -x \
linux-dtb-current-sunxi_2.1.0_armhf.deb .
test@test:~/wukongpi_build/output/debs/test$ ls
boot       linux-image-current-sunxi_2. 1.0_armhf.deb       usr
```

b. 解压后的文件如下所示

```
test@test:~/wukongpi_build/output/debs/test$ tree -L 2
.
├── boot
│   └── dtb-5.4.65-sunxi                          //存放内核使用的 dtb 文件
├── linux-dtb-current-sunxi_2. 1.0_armhf.deb
└── usr
    └── share

4 directories, 1 file
```

13) wukongpi-bulid 编译系统编译 linux 内核源码时首先会将 linux 内核源码和 github 服务器的linux 内核源码进行同步，所以如果想修改linux 内核的源码，首先需要关 闭源码的更新功能（**需要完整编译过一次 linux 内核源码后才能关闭这个功能，否 则会提示找不到linux 内核的源码**），否则所作的修改都会被还原，方法如下：

设置 userpatches/config-default.conf 中的 IGNORE_UPDATES 变量为“yes”

```
test@test:~/wukongpi-build$ vim userpatches/config-default.conf
IGNORE_UPDATES="yes"
```

14) 如果对内核做了修改了，可以使用下面的方法来更新开发板 linux 系统的内核和 内核模块

a. 将编译好的 linux 的 deb 包上传到开发板的 linux 系统中

```
test@test:~/wukongpi-build$ cd output/debs
test@test:~/wukongpi_build/output/debs$ scp \
linux-image-current-sunxi_2.1.0_armhf.deb root@192.168.1.207:/root
```

b. 然后登录到开发板，卸载已安装的 u-boot 的 deb 包

```
root@wukongpi:~# apt purge -y linux-image-current-sunxi
```

c. 再安装刚才上传的新的 u-boot 的 deb 包

```
root@wukongpi:~# dpkg -i linux-image-current-sunxi_2.1.0_armhf.deb
```

d. 然后重启开发板，再查看内核相关的修改是否已生效

15) 安装内核头文件到 linux 系统中的方法如下所示

a. 将编译好的 linux 头文件的 deb 包上传到开发板的 linux 系统中

```
test@test:~/wukongpi-build$ cd output/debs
test@test:~/wukongpi-build/output/debs$ scp \
linux-headers-current-sunxi_2.1.0_armhf.deb root@192.168.1.207:/root
```

b. 然后登录到开发板，安装刚才上传的的 linux 头文件的deb 包

```
root@wukongpi:~# dpkg -i linux-headers-current-sunxi_2.1.0_armhf.deb
```

c. 安装完后在/usr/src 中就可以看到刚才安装的内核头文件相关的内容了

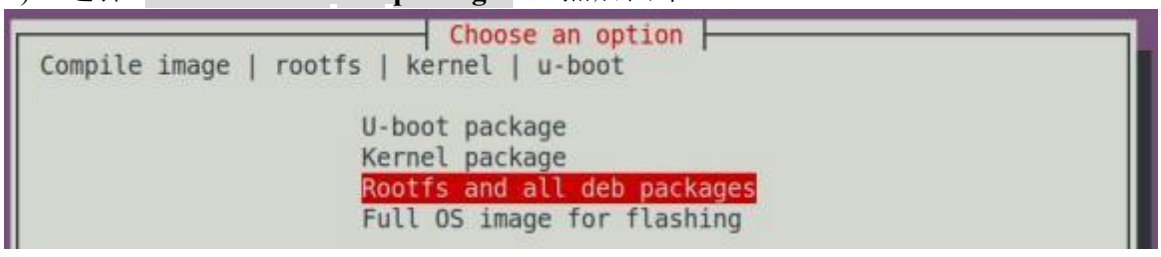
```
root@wukongpi:~# ls /usr/src
linux-headers-current-sunxi
root@wukongpi:~# ls /usr/src/linux-headers-current-sunxi
Documentation Module.symvers certs firmware init lib
net security usr Kconfig arch crypto fs ipc mm
samples sound virt Makefile block drivers include kernel
modules scripts tools
```

4.4. 编译 rootfs

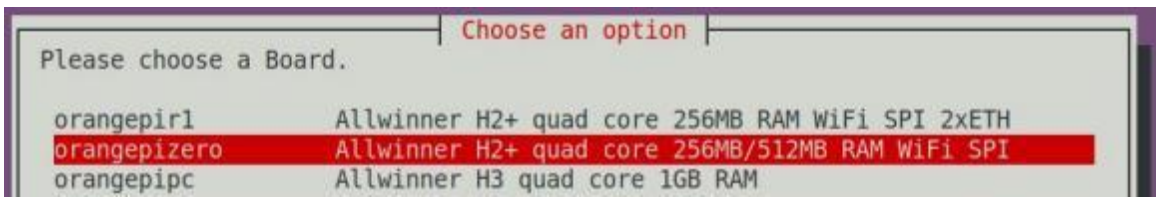
1) 运行 build.sh 脚本，记得加 sudo 权限

```
test@test:~/wukongpi-build$ sudo ./build.sh
```

2) 选择 **Rootfs and all deb packages** ，然后回车



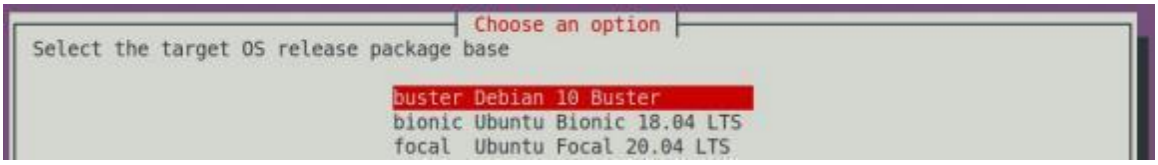
3) 接着选择开发板的型号



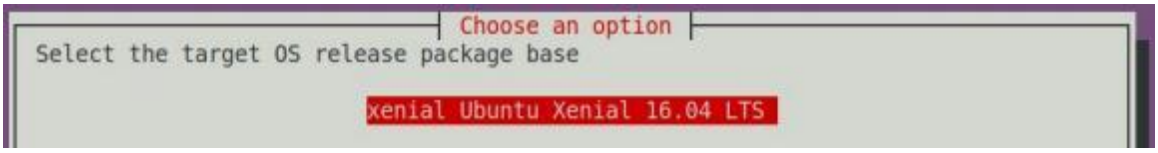
4) 然后选择 rootfs 的类型

buster	Debian 10
bionic	Ubuntu 18.04
focal	Ubuntu 20.04
xenial	Ubuntu16.04

a. linux5.4 支持的 linux 发行版如下所示



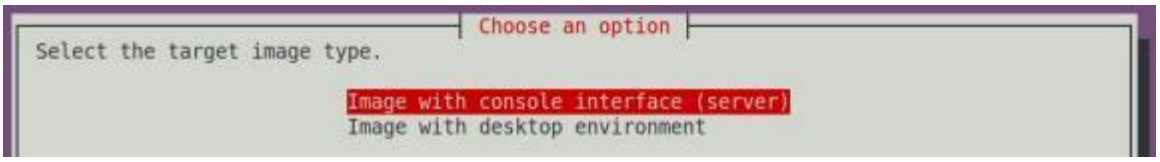
b. linxu3.4 支持的 linux 发行版如下所示



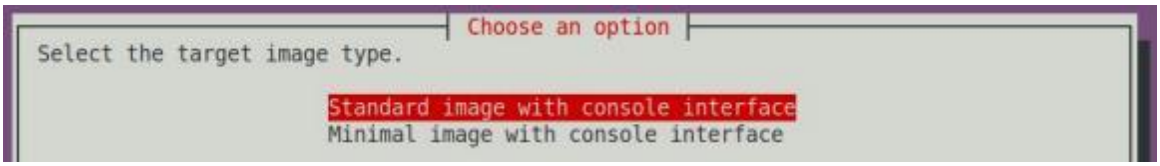
5) 然后选择镜像的类型

a. **Image with console interface** 表示服务器版的镜像，体积比较小

b. **Image with desktop environment** 表示带桌面版镜像，体积比较大



6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多



7) 选择镜像的类型后就会开始编译 rootfs ，编译时会提示下面的信息

a. rootfs 的类型

[o.k.] local not found [Creating new rootfs cache for **bionic**]

b. 编译生成的 rootfs 压缩包的存放路径

[o.k.] Target directory [**external/cache/rootfs**]

a. 编译生成的 rootfs 压缩包的名字

[o.k.] File name [**bionic-cli-armhf.153618961f14c28107ca023429aa0eb9.tar.lz4**]

b. 编译使用的时间

```
[ o.k. ] Runtime [ 13 min ]
```

c. 重复编译 rootfs 的命令，使用下面的命令无需通过图形界面选择，可以直接 开始编译 rootfs

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=wukongpizero
BRANCH=current BUILD_OPT=rootfs RELEASE=bionic
BUILD_MINIMAL=no BUILD_DESKTOP=no
KERNEL_CONFIGURE=yes ]
```

8) 查看编译生成的 rootfs 压缩包

a. bionic-cli-armhf. 153618961f14c28107ca023429aa0eb9.tar.lz4 是 rootfs 的压缩 包，名字各字段的含义为

a) **bionic** 表示 rootfs 的 linux 发行版的类型

b) **cli** 表示 rootfs 为服务器版的类型，如果为 **dekstop** 则表示桌面版类型

c) **armhf** 表示 rootfs 的架构类型

d) 153618961f14c28107ca023429aa0eb9 是由 rootfs 安装的所有软件包的包 名生成的 MD5 哈希值，只要没有修改 rootfs 安装的软件包的列表，那 么这个值就不会变，编译脚本会通过这个 MD5 哈希值来判断是否需要 重新编译 rootfs

b. bionic-cli-armhf. 153618961f14c28107ca023429aa0eb9.tar.lz4.list 列出了 rootfs

安装的所有软件包的包名

```
test@test:~/wukongpi_build$ ls external/cache/rootfs/
bionic-cli-armhf.153618961f14c28107ca023429aa0eb9.tar.lz4
bionic-cli-armhf.153618961f14c28107ca023429aa0eb9.tar.lz4.list
```

9) 如果需要的 rootfs 在 **external/cache/rootfs** 下已经存在，那么再次编译 rootfs 就会直接跳过编译过程， 不会重新开始编译， 编译镜像的时候也会去 **external/cache/rootfs** 下查找是否已经有缓存可用的 rootfs ，如果有就直接使用，这样可以节省大量的下载编译时间

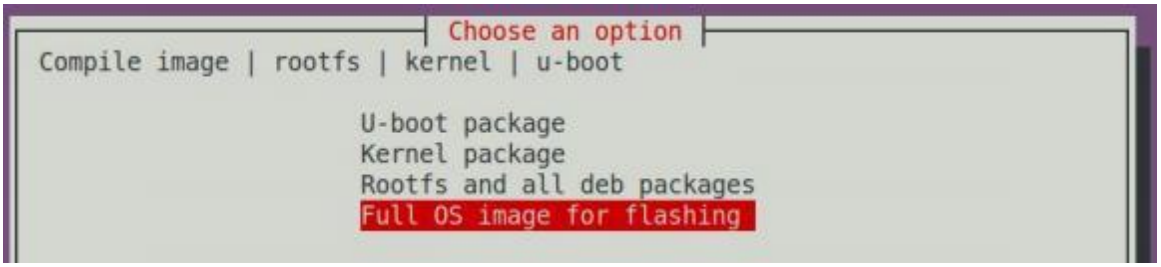
10) 由于编译 rootfs 的时间较长，如果不想从头开始编译 rootfs ，或者编译 rootfs 的 过程有问题，可以直接下载 Wukong Pi 缓存的 rootfs 压缩包，rootfs 压缩包百度云盘 的下载链接如下所示，下载好的 rootfs 压缩包(不要解压哦)需要放在 wukongpi-build的external/cache/rootfs 目录下才能被编译脚本正常使用

4.5. 编译 linux 镜像

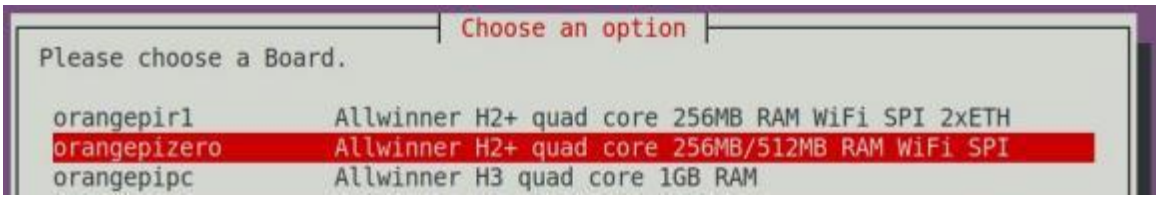
1) 运行 build.sh 脚本，记得加 sudo 权限

```
test@test:~/wukongpi-build$ sudo ./build.sh
```

2) 选择 **Full OS image for flashing** ，然后回车



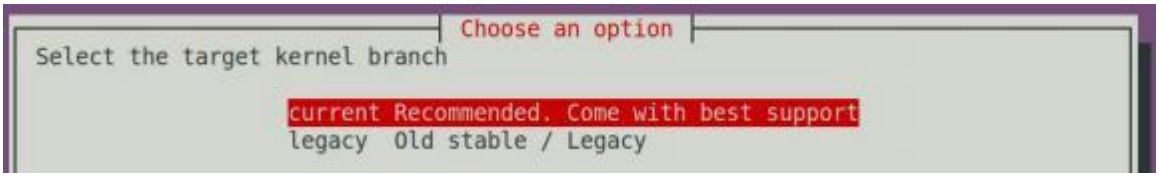
3) 然后选择开发板的型号



4) 然后选择分支

a. current 会编译 u-boot v2020.04 、linux5.4

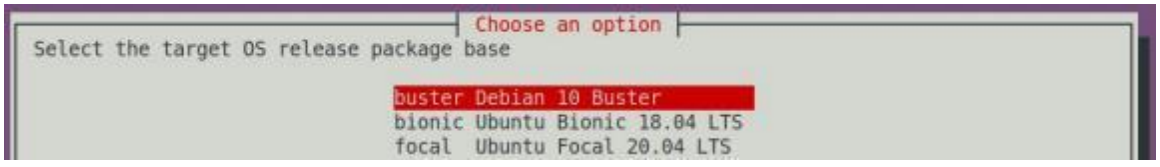
b. legacy 会编译 u-boot v2018.05 、linux3.4



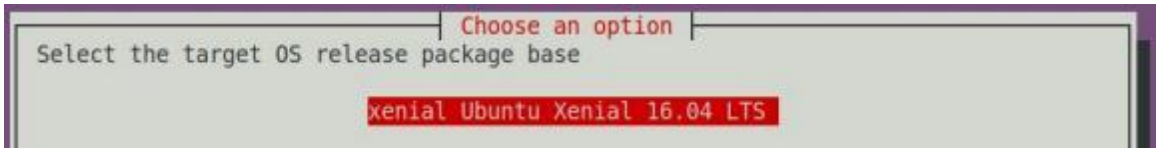
5) 然后选择 rootfs 的类型

buster	Debian 10
bionic	Ubuntu 18.04
focal	Ubuntu 20.04
xenial	Ubuntu16.04

a. linux5.4 支持的 linux 发行版如下所示



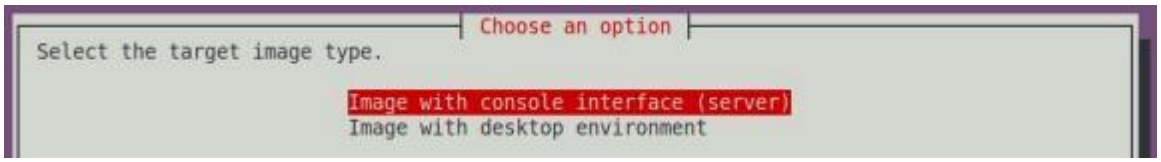
b. linxu3.4 支持的 linux 发行版如下所示



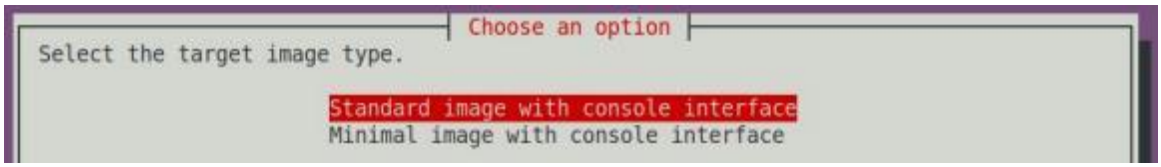
6) 然后选择镜像的类型

a. **Image with console interface** 表示服务器版的镜像，体积比较小

b. **Image with desktop environment** 表示带桌面版镜像，体积比较大



7) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多



8) 选择镜像的类型后就会开始编译 linux 镜像，编译镜像的大致流程如下

a. 编译 u-boot 源码，生成 u-boot 的 deb 包

b. 编译 linux 源码，生成 linux 相关的 deb 包

c. 制作 linux firmware 的 deb 包

d. 制作 wukongpi-config 工具的 deb 包

e. 制作板级支持的deb 包

f. 如果是编译 desktop 版镜像，还会制作 desktop 相关的 deb 包

g. 检查 rootfs 是否已经缓存，如果没有缓存，则重新制作 rootfs ，如果已经缓 存，则直接解压使用

h. 安装前面生成的 deb 包到 rootfs 中

i. 对不同的开发板和不同类型镜像做一些特定的设置，如预装额外的软件包， 修改配置文件

j. 然后制作镜像文件，并格式化分区，默认类型为 ext4

k. 再将配置好的 rootfs 拷贝到镜像的分区中

l. 然后更新 initramfs

m. 最后将 u-boot 的 bin 文件通过 dd 命令写入到镜像中

9) 编译完镜像后会提示下面的信息

a. 编译生成的 linux 镜像的存放路径

```
[ o.k. ] Done building
[ output/images/wukongpizero_2.1.0_ubuntu_bionic_server_linux5.4.65/wukongpizero _2.1.0_ubuntu_bionic_server_linux5.4.65.img ]
```

b. 编译镜像使用的时间

```
[ o.k. ] Runtime [ 9 min ]
```

c. 重复编译镜像的命令，使用下面的命令无需通过图形界面选择，可以直接开 始编译镜像

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=wukongpizero BRANCH=current BUILD_OPT=image RELEASE=bionic BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```