飞腾派 OS

用户使用开发手册

（V1.2）

飞腾信息技术有限公司
www.phytium.com.cn
2024 年 05 月 08 日

商标声明

Phytium 和其他飞腾商标均为飞腾信息技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

特别提示

本文档仅作为使用指导，飞腾对本文档内容不做任何明示或暗示的声明或保证。本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

由于产品版本升级或其他原因，本文档内容会不定期进行更新，如有变更，恕不另行通知。

最新技术资源

飞腾嵌入式软件开源社区 https://gitee.com/phytium_embedded

或者访问飞腾软件开发者平台 https://service.phytium.com.cn/developer/

目录

1 飞腾派 OS 简介 3
1.1 飞腾派开发板的硬件规格 3
1.2 飞腾派 OS 的主要功能 4
1.3 支持的主要驱动 4
1.4 支持的主要软件包 5
2 飞腾派 OS 使用指南 6
2.1 准备 SD 卡启动镜像 6
2.1.1 Windows 烧录镜像 6
2.1.2 Linux 烧录镜像 6
2.2 系统启动 6
2.3 桌面基本介绍 7
2.3.1 web 浏览器 7
2.3.2 终端 8
2.3.3 文件管理器 8
2.4 网络连接设置 9
2.4.1 有线网络 9
2.4.2 WiFi 网络 9
2.5 蓝牙设置 10
2.6 使用 SSH 12
2.7 使用调试串口 13
2.8 软件安装 14
3 飞腾派 OS 开发指南 15
3.1 构建飞腾派 OS 15
3.1.1 下载 phytium-pi-os 16
3.1.2 基本 defconfig 16
3.1.3 编译 SD 卡镜像 16
3.1.4 phytiumpi_ethercat config 16
3.1.5 phytiumpi_xenomai config 16

3.1.6 phytium_optee config 17
3.1.7 openamp config 17
3.1.8 清理编译结果 18
3.2 Buildroot 使用技巧 18
3.2.1 树外构建 18
3.2.2 内核源码相关的修改与内核替换 18
3.2.3 二次编译减少编译时间 19
3.3 使用新内核 19
3.3.1 交叉编译内核 19
3.3.2 飞腾派开发板上编译内核 20
3.3.3 启动新内核 21
3.4 编译内核模块 23
3.4.1 交叉编译内核模块 23
3.4.2 飞腾派开发板上编译内核模块 23
3.5 Buildroot 编译新的应用软件 24
3.5.1 Buildroot 软件包介绍 24
3.5.2 编写 Buildroot 软件包 24
3.5.3 编译软件包 25
A 更新记录 25
B 扩展资料 25

1 飞腾派 OS 简介

飞腾派开发板是一款面向广大工程师和爱好者的开源硬件。主板处理器采用飞腾四核处理器,兼容 ARM V8 指令集,主频最高可达 1.8GHz。

飞腾派 OS（Phytium Pi OS）是运行在飞腾派开发板上的操作系统，基于 Debian 并针对飞腾派开发板深度定制，其中包括固件，内核，各接口驱动程序，桌面系统，用于各种场景的开发包等。

它可以烧录在 SD 卡中，作为飞腾派开发板的启动系统。

1.1 飞腾派开发板的硬件规格

开发板内置 2/4GB DDR4 内存，双路千兆以太网、USB、UART、CAN、HDMI、音频等接口。主板板载 WiFi 蓝牙,陶瓷天线,可快速连接无线通信。集成一路 miniPCIE 接口，可实现 AI 加速卡与 4G 通信等。

功能	描述
CPU	飞腾四核处理器,ARMV8 架构
内存	2G、4G 版本,64 位 DDR4
存储	支持 microSD 和 EMMC 启动，默认 microSD
网络	2×千兆以太网(RJ45)
USB	1×USB3.0 host,3×USB2.0 host
PCIe	1×Mini-PCIe ,支持 4G、AI 等模组
蓝牙	板载蓝牙 BT4.2/ BLE4.2
WiFi	板载 2.4G + 5G 双频 WIFi
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(可配置为 UART 模式)
I2C	2+2×MIO(可配置为 I2C 模式)
I2S	1 路
SPI	2 路
CAN	2 路 CANFD
GPIO	最多 29 个

供电	12V3A 直流电源
工作温度	0~50° C

1.2 飞腾派 OS 的主要功能

linux 5.10.209 内核

多种外设驱动

Debian 11 定制系统

Xfce 桌面系统

支持 WiFi6 双频配置

支持蓝牙 配置

支持 4G/5G 网卡

支持 AI 加速卡

支持 web，Python 开发包

支持类树莓派 HAT 板开发包

支持 EtherCAT+ Linux RT

支持 Xenomai

支持 Phytium OP-TEE

支持 OpenAMP

1.3 支持的主要驱动

mirco SD

eMMC

千兆以太网控制器及收发器

USB

miniPCIe

蓝牙 v4.2

wifi6

UART

I2C

I2S

SPI

CAN

GPIO

MIO

watchdog

1.4 支持的主要软件包

openssl

ffmpeg

ssh

gdb

Python3

lua5.1

curl

nfs-common

ntfs-3g

cifs-utils

mkvtoolnix

wpasupplicant

wireless-tools

dhcpd5

vlc

bluez

pulseaudio

blueman

bluetooth

locales

2 飞腾派 OS 使用指南

2.1 准备 SD 卡启动镜像

下载 SD 卡的镜像 sdcard.img，可以在 Linux 或 Windows 上将镜像烧录到 SD 卡中。请准备一张容量大于 16G 的 SD 卡，并连接到 Linux 或 Windows 主机上。

2.1.1 Windows 烧录镜像

使用 win32 Disk imager 进行烧录。

具体使用方法见 https://blog.csdn.net/Mr_LanGX/article/details/123338081

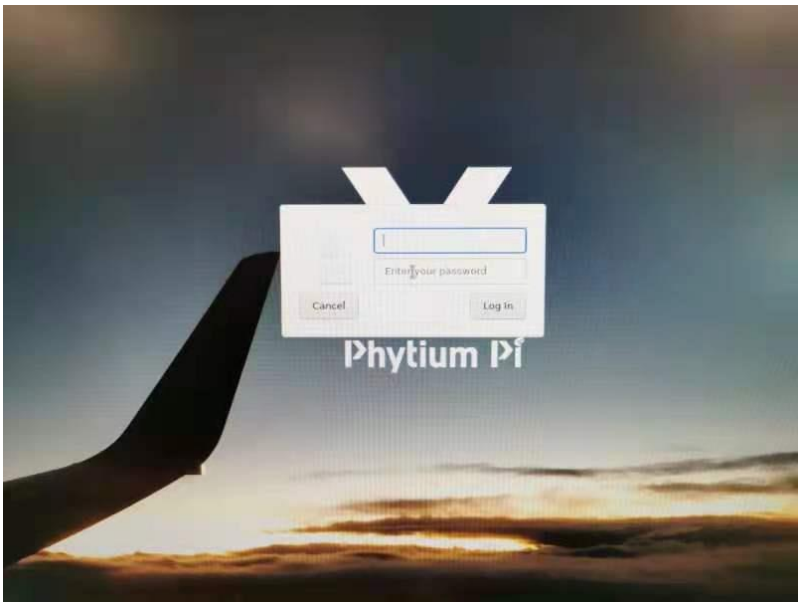
2.1.2 Linux 烧录镜像

主机端将 SD 卡接入主机（以主机识别设备名为/dev/sdb 为例，请按实际识别设备名更改，确定设备没有被挂载，如果有挂载,需要 umount）

```
$ sudo dd if=sdcard.img of=/dev/sdb bs=1M status=progress
```

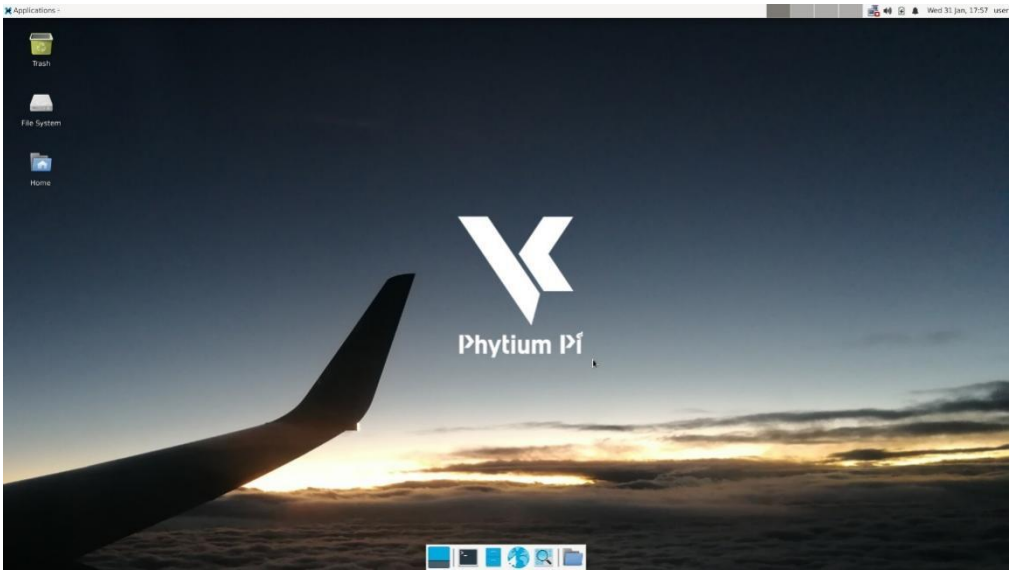
2.2 系统启动

将有镜像的 SD 卡插入（确认开发板配置为 SD 卡启动，参见飞腾派硬件规格书 7.13 节“启动选择开关”），HDMI 连接显示器，连接 USB 键盘鼠标，连接电源后系统就会自动启动。之后，在显示器上会出现小企鹅标志，最后显示登录界面。登录的用户名：user，密码：user



2.3 桌面基本介绍

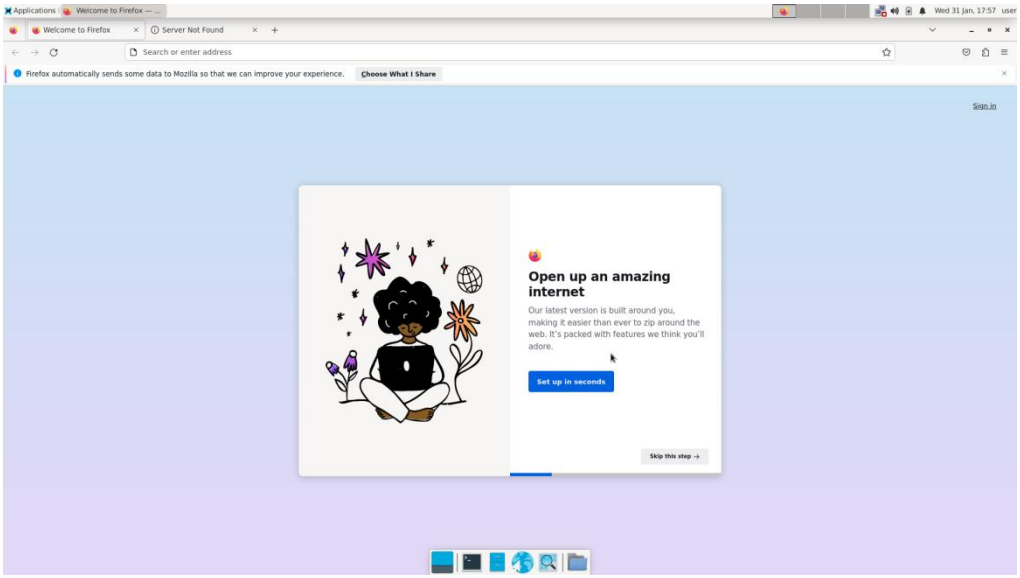
Xfce 作为默认的桌面环境,是一款轻量级桌面环境，注重效率和资源利用，它提供了简洁而直观的界面。



在桌面底部中间的位置，从左到右提供 “显示桌面”，“终端”，“文件管理器”，“web 浏览器”，“程序查找器” “快捷文件管理器” 。

2.3.1 web 浏览器

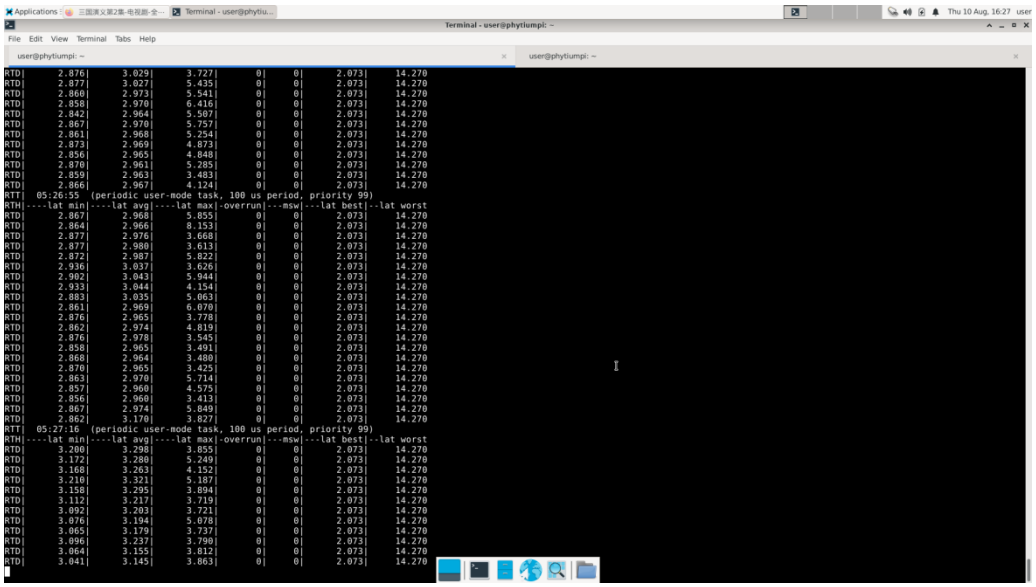
Xfce 提供 firefox 作为默认浏览器



2.3.2 终端

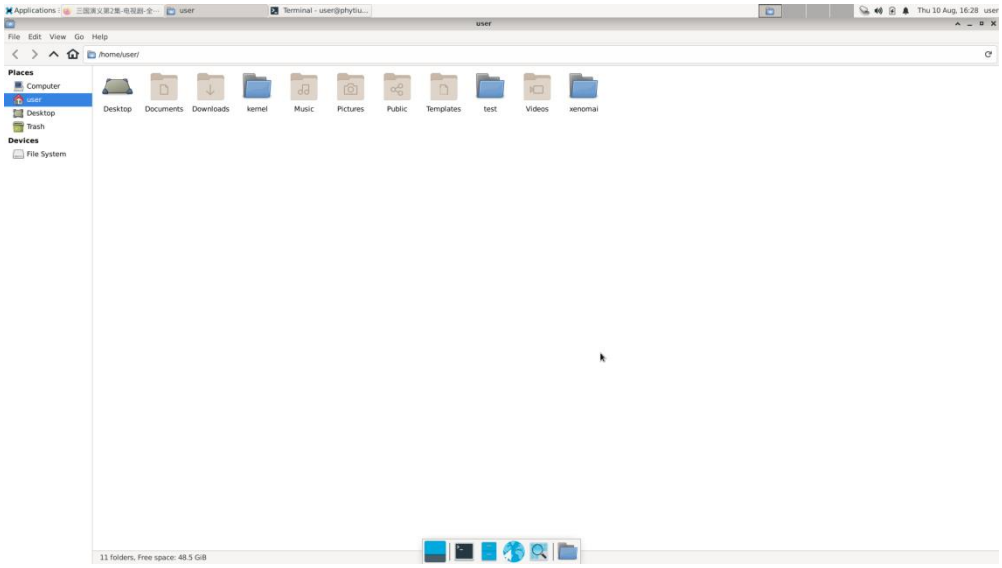
在终端中，可以使用命令行来管理和设置系统，开发软件。

user 用户可以在终端中 sudo 来获取 root 权限。



2.3.3 文件管理器

通过文件管理器，可以管理文件，打开，修改，保存文件。

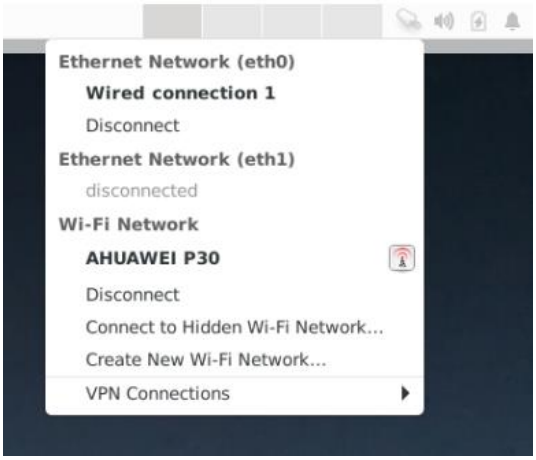


2.4 网络连接设置

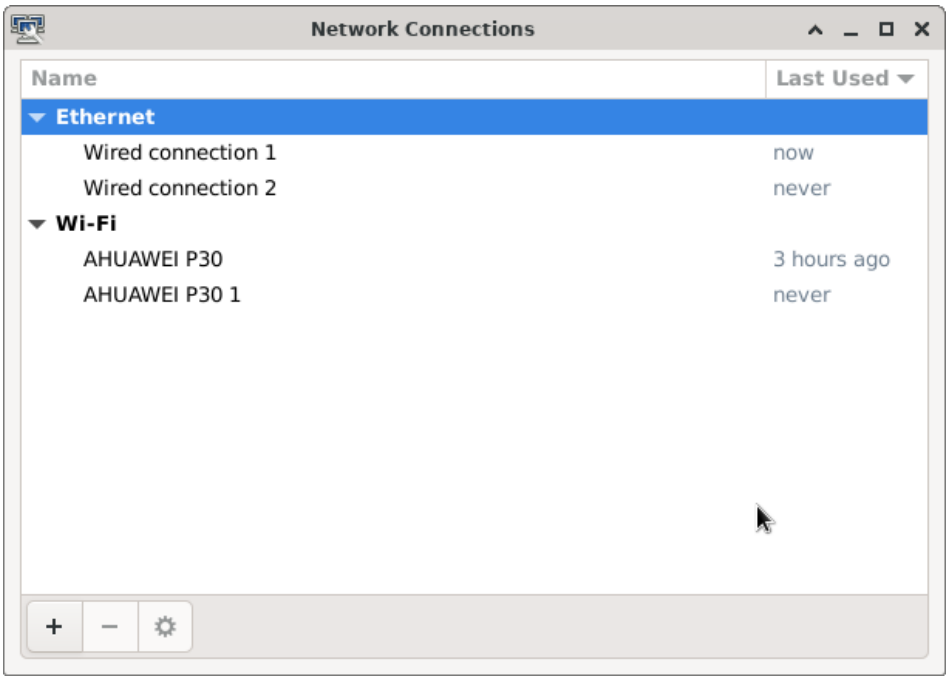
2.4.1 有线网络

默认两个网口都是 DHCP 的，可以连接路由器获取 IP 地址。

点击桌面右上角的连接图标来查看有线网网络连接情况。



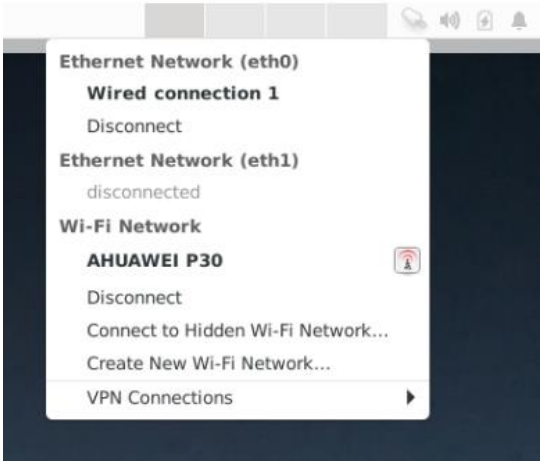
通过网络管理器来设置你的网络参数，增加删除连接，修改连接参数。



2.4.2 WiFi 网络

系统已经集成 WIFI 驱动. 启动后可以自动加载看到 wlan0。 wpasupplicant 服务默认自动启动。

点击桌面右上角的连接图标来查看 WiFi 网络。



点击 create new WiFi network，来创建一个连接，输入 network name 和 key，就可以连接到 WiFi 网络。

禁止 WiFi 服务在开机时自动运行

```
$ sudo systemctl disable wpa_supplicant.service
$ sudo reboot
```

开启 WiFi 服务在开机时自动运行

```
$ sudo systemctl enable wpa_supplicant.service
$ sudo reboot
```

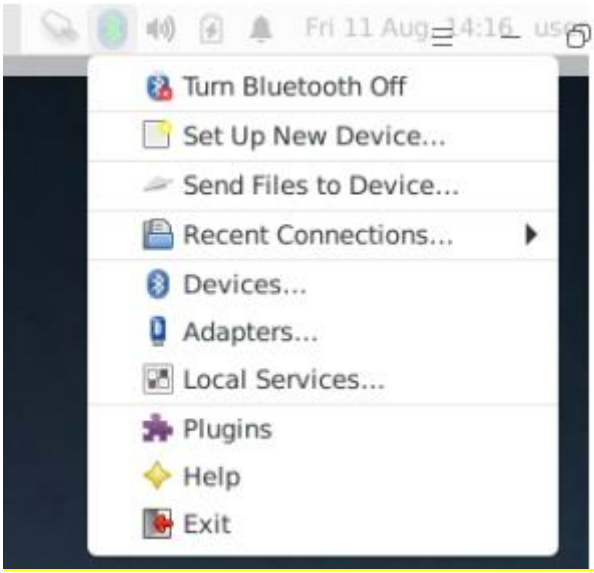
2.5 蓝牙设置

蓝牙服务系统默认是关闭的。启动步骤如下：

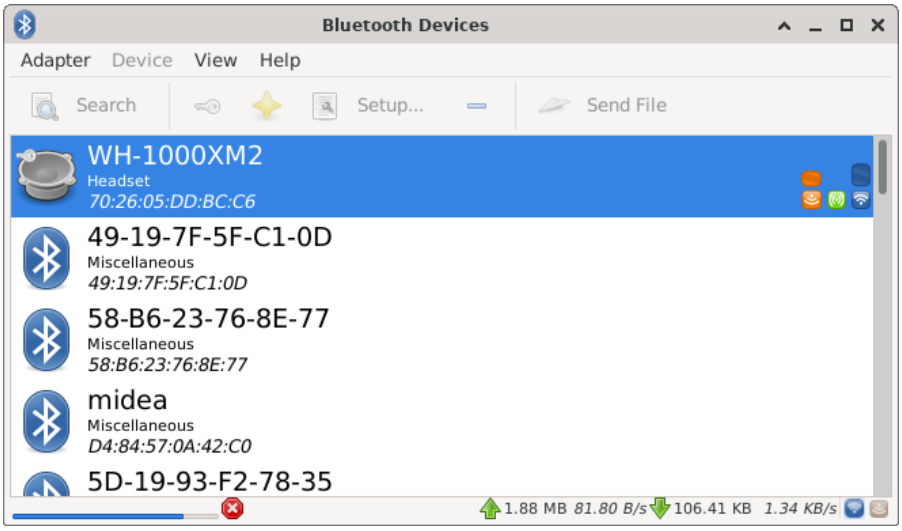
```
$ sudo systemctl enable bluetooth.service
```

```
$ sudo systemctl enable systemd-hciattach.service
$ sudo reboot
```

重启后，在桌面右上角的状态栏中有蓝牙图标，点击图标可以添加蓝牙设备。

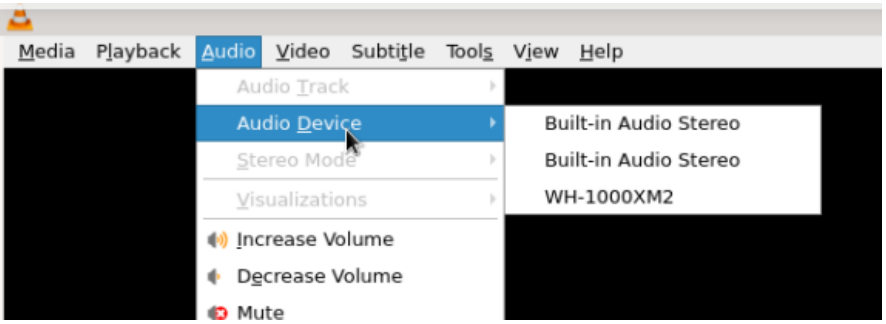


在添加设备窗口，点击“search”找到要连接的蓝牙耳机和音响，右键选连接。



如果通过蓝牙播放音乐

vlc 播放器选择 Audio —-> Audio Device -> <蓝牙耳机设备



2.6 使用 SSH

SSH client 和 SSH server 默认已经集成在飞腾派 OS 中，可以直接使用。

首先确保飞腾派有线网口或 wifi 网口有 IP 地址（下面的例子中 eth0 的 ip address 是 10.10.80.174），可以通过 ip addr 命令查看。

```
user@phytiumpi:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 6c:b3:11:0f:9c:e8 brd ff:ff:ff:ff:ff:ff permaddr 00:11:22:33:44:55
    inet 10.10.80.174/24 brd 10.10.80.255 scope global dynamic noprefixroute eth0
       valid_lft 1263sec preferred_lft 1263sec
    inet6 fe80::8cf3:ea81:564c:a144/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 10:22:33:44:55:66 brd ff:ff:ff:ff:ff:ff
```

其次通过 ping 命令查看和远端设备的连通性。（飞腾派 IP 10.10.80.174，远端设备 IP 10.10.80.170）

```
$ ping 10.10.80.174
PING 10.10.80.174 (10.10.80.174) 56(84) bytes of data.
64 bytes from 10.10.80.174: icmp_seq=1 ttl=64 time=0.350 ms
64 bytes from 10.10.80.174: icmp_seq=2 ttl=64 time=0.294 ms
^C
--- 10.10.80.174 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1028ms
rtt min/avg/max/mdev = 0.294/0.322/0.350/0.028 ms
```

最后在确认可以 ping 通远端设备的情况下：

a. 从飞腾派开发板登录远程设备

```
user@phytiumpi:~$ ssh phytium@10.10.80.170
The authenticity of host '10.10.80.170 (10.10.80.170)' can't be established.
ECDSA key fingerprint is
```

```
SHA256:C7MVQ7GzEb4EKkINKnrEfmQ/Cu0VkpRzypMbeHgpJEw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.80.170' (ECDSA) to the list of known hosts.
phytium@10.10.80.170's password:
Welcome to Kylin V10 SP1 (GNU/Linux 5.4.18-57-generic aarch64)

 * Management:        http://www.kylinos.cn/ * Support:
http://www.kylinos.cn/service.aspx
Last login: Fri Jun    9 15:08:07 2023 from 10.10.80.177
phytium@phytium-d2000:~$
```

b. 从远程设备登录飞腾派开发板 （ password: user ）

```
$ ssh user@10.10.80.174
The authenticity of host '10.10.80.174 (10.10.80.174)' can't be established.
ECDSA key fingerprint is
SHA256:6CzIeVEaWgskwMdXDu/VcnS6c85/S/O4fUWaHdTVJY0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.80.174' (ECDSA) to the list of known hosts.
user@10.10.80.174's password:
Linux phytiumpi 5.10.209-rt101-phytium-embeded-v2.1 #1 SMP PREEMPT_RT Tue
Aug 8 14:09:59 CST 2023 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Aug    9 18:13:43 2023 from 10.10.80.198
user@phytiumpi:~$
```

2.7 使用调试串口

串口连接请参考《飞腾派_V2 硬件规格书》中的调试串口位置（管脚 8，10，12），连接串口。

串口终端程序必须设置为：

波特率:115200

数据位:8

奇偶校验:无

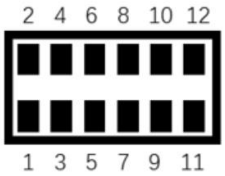
停止位:1

系统调试串口下的用户有 user 和 root

用户名:user;密码:user

用户名:root;密码:root

多功能接口 2：包含调试串口和 CAN 总线接口，具体定义如下。



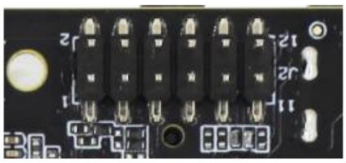


图 17 功能接口 2

表 6 多功能接口 2 定义

脚标	信号	脚标	信号
1	VCC 1.8V	2	CAN0_TX
3	GPIO2_10	4	CAN0_RX
5	GPIO1_9	6	VCC 3.3V
7	GPIO1_10	8	DEBUG_UART1_TXD
9	GPIO0_5	10	DEBUG_UART1_RXD
11	V_BAT	12	GND

2.8 软件安装

可以在命令行终端中通过 apt-get install 来安装，或者在图形界面中双击.deb 包文件进行安装。

其他的 Debian 使用问题可以参考如下文档：

https://www.debian.org/doc/manuals/debian-reference/

3 飞腾派 OS 开发指南

3.1 构建飞腾派 OS

飞腾派 OS 是由 Buildroot 生成的，它是一种简单、高效且易于使用的工具，可以通过交叉编译在主机系统上生成飞腾派 OS 的镜像文件。关于更多 Buildroot 的信息，可以参考 https://buildroot.org/downloads/manual/manual.html

我们只支持在 Ubuntu20.04、Ubuntu22.04、Debian11 这三种 x86 主机上运行 Buildroot，不支持其他系统。首先，Buildroot 需要主机系统上安装如下 Linux 程序，请检查是否已安装：

```
Build tools:
- which
- sed
- make (version 3.81 or any later)
- binutils
- build-essential (only for Debian based systems)
- gcc (version 4.8 or any later)
- g++ (version 4.8 or any later)
- bash
- patch
- gzip
- bzip2
- perl (version 5.8.7 or any later)
- tar
- cpio
- unzip
- rsync
- file (must be in /usr/bin/file)
- bc
Source fetching tools:
- wget
- git
```

除此之外，还需要安装如下软件包：

```
$ sudo apt install debootstrap qemu-user-static binfmt-support debian-archive-keyring
```

对于 Debian11 系统，需要设置 PATH 环境变量：

```
$ PATH=$PATH:/usr/sbin
```

3.1.1 下载 phytium-pi-os

```
$ git clone https://gitee.com:phytium_embedded/phytium-pi-os.git
```

3.1.2 基本 defconfig

飞腾派构建的文件系统的配置文件位于 configs 目录。

在 phytium-pi-os 根目录下执行$ make list-defconfigs，返回 configs 目录中的 defconfig 配置文件。

```
$ make list-defconfigs
```

其中以 phytiumpi 开头的为飞腾派相关的 defconfig 配置文件，包含：

```
phytiumpi_defconfig                    - Build for phytiumpi (without desktop)
phytiumpi_desktop_defconfig             - Build for phytiumpi_desktop
```

3.1.3 编译 SD 卡镜像

（1）加载 defconfig

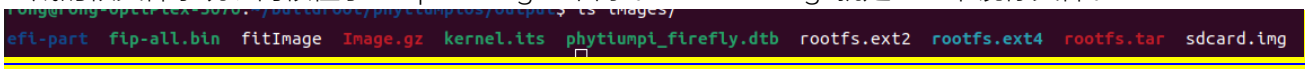
```
$ make phytiumpi_xxx_defconfig
```

（2）编译

```
$ make
```

（3）镜像的输出位置

生成的根文件系统、内核位于 output/images 目录。 sdcard.img 就是 SD 卡镜像文件。



3.1.4 phytiumpi_ethercat config

编译 EtherCAT，它的信息请参考：https://gitee.com/phytium_embedded/ether-cat

支持将 EtherCAT 驱动及用户态的库、工具编译安装到 Debian 系统上。如果需要编译请执行：

（1）使用 phytiumpi_xxx_defconfig 作为基础配置项，在 phytiumpi_xxx_defconfig 中加入支持 rt 内核，及 ethercat 的配置：

```
$ ./support/kconfig/merge_config.sh configs/phytiumpi_xxx_defconfig
configs/phytiumpi_linux_5.10_rt.config configs/phytiumpi_ethercat.config
```

（2）编译

```
$ make
```

3.1.5 phytiumpi_xenomai config

编译 Xenomai，它的信息请参考：

https://gitee.com/phytium_embedded/linux-kernel-xenomai

支持将 Xenomai 内核及用户态的库、工具编译安装到飞腾派 OS 上。如果需要编译请执行：

（1）使用 phytiumpi_xxx_defconfig 作为基础配置项，在 phytiumpi_xxx_defconfig 加入 xenomai 的配置：

```
$ ./support/kconfig/merge_config.sh configs/phytiumpi_xxx_defconfig
configs/phytiumpi_xenomai_xxx.config
```

phytiumpi_xenomai_xxx.config 为以下配置片段文件之一：

```
phytiumpi_xenomai_mercury_5.10.config （linux 5.10 rt 内核+xenomai-v3.2.2.tar.gz）
phytiumpi_xenomai_cobalt_5.10.config （xenomai cobalt 5.10 内核+xenomai-
v3.2.2.tar.gz）
```

（2）编译

```
$ make
```

3.1.6 phytium_optee config

编译 Phytium OP-TEE，它的信息请参考：

https://gitee.com/phytium_embedded/phytium-optee

如果需要编译请执行：

（1）使用 phytiumpi_xxx_defconfig 作为基础配置项，合并支持 OP-TEE 的配置：

```
$ ./support/kconfig/merge_config.sh configs/phytiumpi_xxx_defconfig
configs/phytiumpi_optee.config
```

（2）编译

```
$ make
```

3.1.7 openamp config

编译 OpenAMP，它的信息请参考：

https://gitee.com/phytium_embedded/phytium-standalone-sdk

https://gitee.com/phytium_embedded/phytium-free-rtos-sdk

如果需要编译请执行：

（1）使用 phytiumpi_xxx_defconfig 作为基础配置项，合并支持 OpenAMP 的配置：

```
$ ./support/kconfig/merge_config.sh configs/phytiumpi_xxx_defconfig
configs/openamp_xxx.config
```

openamp_xxx.config 为以下配置片段文件之一：

```
openamp_standalone.config （OpenAMP 裸跑二进制镜像的相关配置）
openamp_free_rtos.config （OpenAMP FreeRTOS 二进制镜像的相关配置）
```

（2）编译

```
$ make
```

3.1.8 清理编译结果

（1）make clean

删除所有编译结果，包括 output 目录下的所有内容。当编译完一个文件系统后，编译另一个文件系统前，需要执行此命令。

（2）make distclean

重置 Buildroot，删除所有编译结果、下载目录以及配置。

3.2 Buildroot 使用技巧

3.2.1 树外构建

树外构建可以实现并行运行多个构建，以下三种方式均可实现树外构建：

（1） 在源码根目录下，使用 O 变量指定输出目录：

```
$ make O=xxx/foo-output phytiumpi_xxx_defconfig
```

（2） 在一个空的输出目录运行，指定 O 变量和源码根目录的路径：

```
$ mkdir -p xxx/foo-output
$ cd xxx/foo-output
$ make -C xxx/phytium-pi-os/ O=$(pwd) phytiumpi_xxx_defconfig
```

（3） 对于使用 merge_config.sh 合并配置文件的情况，在源码目录运行：

```
$ mkdir -p xxx/foo-output
$ ./support/kconfig/merge_config.sh -O xxx/foo-output configs/phytiumpi_xxx_defconfig
configs/phytiumpi_xxx.config
```

编译前，需要进入上述指定的输出目录，执行：

```
$ make
```

3.2.2 内核源码相关的修改与内核替换

默认的内核源码放在output/build/linux-<version>，如果在该目录对内核源码进行修改，当运行make clean 后该目录会被删除，修改也就丢失了。

因此，Buildroot 提供了<PKG>_OVERRIDE_SRCDIR 机制。

首先，需要创建 local.mk 文件。该文件要与.config 文件放在一个目录下（对于树内构建是源码根目录，对于树外构建是树外构建的输出目录）。在 local.mk 文件键入以下内容：

```
$ LINUX_OVERRIDE_SRCDIR = path/to/kernel
```

在 path/to/kernel 目录下修改内核源码，之后执行

```
$ make linux-rebuild phyuboot-rebuild all
```

编译出修改后的内核并生成最新的 SD 卡镜像。

如果用户不想重新烧录 SD 卡镜像，只想在已启动的飞腾派开发板上直接替换上述已修改的内核，请将上述编译出的 output/images/fitImage 文件拷贝至飞腾派开发板某个目录下，并在开发板该目录下执行：

```
$ sudo runtime_replace_bootloader.sh image
```

3.2.3 二次编译减少编译时间

一次完整的编译过程消耗了大量的时间在下载源码上，可以将上次编译时下载的源码 tar 包备份，以节省下载源码的时间。

具体做法是：将 dl 目录备份，在 make clean && make distclean 之后，将备份的 dl 目录拷贝到 phytium-pi-os 下。

3.3 使用新内核

3.3.1 交叉编译内核

如果用户想自己手工命令构建内核，设备树以及内核模块，可以按照该章节操作。

在 x86 交叉编译 arm64 内核，推荐使用编译器的链接如下：

https://developer.arm.com/-/media/Files/downloads/gnu-a/10.2-2020.11/binrel/gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu.tar.xz

将其解压缩到/opt 目录后，然后设置环境变量，操作步骤如下：

```
$ export PATH=/opt/gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu/bin:$PATH
$ export ARCH=arm64 CROSS_COMPILE=aarch64-none-linux-gnu-
$ export CC=aarch64-none-linux-gnu-gcc
```

从如下网址下载 RT 内核源代码（以 RT 内核为例）：

https://gitee.com/phytium_embedded/phytium-linux-kernel/tree/linux-5.10-rt

第一步：进入内核根目录下，按照下述步骤配置和编译 RT 内核

```
$ make phytium_defconfig
$ make
```

其中内核的名称是 Image，设备树的名称是 phytiumpi_firefly.dtb。

第二步：由于内核很多模块编译成 ko，所以需要手工生成 ko 的安装目录和文件：

在内核当前目录下创建 build 目录，作为 ko 的安装目录。当然用户可以设置任何目录为安装目录。

```
$ mkdir build
```

设置 build 目录为安装目录

```
$ export INSTALL_MOD_PATH=`pwd`/build
```

安装模块

```
$ make modules_install
```

查看模块

```
$ ls build/lib/modules/
5.10.209-rt101-phytium-embeded-v2.1
```

其中，模块名称分 2 部分，第一部分“5.10.209-rt101-phytium-embeded”不会变化，第二部分“-v2.1”会随着版本的不断更新而持续更新，本章节后续章节描述内核模块以不变部分为准。

第三步：将 SD 卡插到 USB 读卡器上，然后插到交叉编译机上，假设读卡器的设备节点为 sdc，将内核和设备树安装到 SD 卡的/boot 目录(注意提前备份旧的内核和设备树以及内核模块)。

```
$ sudo mount /dev/sdc1 /mnt
$ sudo cp arch/arm64/boot/Image /mnt/boot/
$ sudo cp arch/arm64/boot/dts/phytium/phytiumpi_firefly.dtb /mnt/boot/
```

然后将配套的内核模块安装到/lib/modules

```
$ sudo cp build/lib/modules/5.10.209-rt101-phytium-embeded  /mnt/lib/modules  -R
$ sudo umount /mnt
```

3.3.2 飞腾派开发板上编译内核

开发板启动注册之后需要安装如下软件，这些软件包如果没有安装，编译内核的时候可能失败。

```
$ sudo apt-get install autoconf automake libtool fuse debhelper findutils autotools-dev
pkg-config libltdl-dev flex bison device-tree-compiler libssl-dev
```

从如下网址下载 RT 内核源代码（以 RT 内核为例）：

https://gitee.com/phytium_embedded/phytium-linux-kernel/tree/linux-5.10-rt

第一步：进入内核根目录下，按照下述步骤配置和编译 RT 内核

```
$ make phytium_defconfig
$ make
```

其中内核的名称是 Image，设备树的名称是 phytiumpi_firefly.dtb。

第二步：由于内核很多模块编译成 ko，所以需要手工生成 ko 的安装目录和文件：

在内核当前目录下创建 build 目录，作为 ko 的安装目录。当然用户可以设置任何目录为安装目录。

```
$ mkdir build
```

设置 build 目录为安装目录

```
$ export INSTALL_MOD_PATH=`pwd`/build
```

安装模块

```
$ make modules_install
```

查看模块

```
$ ls build/lib/modules/
5.10.209-rt101-phytium-embeded-v2.1
```

其中，模块名称分 2 部分，第一部分“5.10.209-rt101-phytium-embeded”不会变化，第二部分“-v2.1”会随着版本的不断更新而持续更新，本章节后续章节描述内核模块以不变部分为准。

第三步：将内核和设备树安装到/boot 目录(注意提前备份旧的内核和设备树以及内核模块)。

```
$ sudo cp arch/arm64/boot/Image /boot/
$ sudo cp arch/arm64/boot/dts/phytium/phytiumpi_firefly.dtb /boot/
```

然后将配套的内核模块安装到/lib/modules

```
$ sudo cp build/lib/modules/5.10.209-rt101-phytium-embeded  /lib/modules  -R
$ sudo umount /mnt
```

与交叉编译内核相比，显然在飞腾派开发板上编译内核上步骤简洁，更换内核也更方便。但是实际上，在飞腾开发板上编译内核速度很慢，效率很低，因为在一般情况下，飞腾派开发板的 CPU 的性能比交叉编译机会差很多。

3.3.3 启动新内核

启动开发板，然后在 Uboot 启动阶段敲击键盘的回车键，这时系统会停留在 Uboot 的 Shell 界面，如下所示。

```
AHCI 0001.0301 32 slots 1 ports 6 Gbps 0x1 impl SATA mode
flags: 64bit ncq stag pm led clo only pmp pio slum part ccc apst
SATA link 0 timeout.
AHCI 0001.0301 32 slots 1 ports 6 Gbps 0x1 impl SATA mode
flags: 64bit ncq stag pm led clo only pmp pio slum part ccc apst
Hit any key to stop autoboot:  0   // 红颜色打印信息回显后键入回车字符
```

```
Phytium-Pi#                                    // 这时进入 Uboot 的 Shell 界面
Phytium-Pi#setenv bootargs console=ttyAMA1,115200 earlycon=pl011,0x2800d000
root=/dev/mmcblk0p1 rootwait rw
Phytium-Pi#ext4load mmc 0:1 0x90100000 /boot/Image
28692992 bytes read in 6293 ms (4.3 MiB/s)
Phytium-Pi#ext4load mmc 0:1 0x90000000 /boot/phytiumpi_firefly.dtb
25125 bytes read in 13 ms (1.8 MiB/s)
Phytium-Pi#booti 0x90100000 - 0x90000000
```

在 Uboot 的 shell 菜单按照如下步骤引导内核和设备树启动。

第一步：设置启动参数，其中，嵌入式 Linux 文件系统

```
Phytium-Pi# setenv bootargs console=ttyAMA1,115200 earlycon=pl011,0x2800d000
root=/dev/mmcblk0p1 rootwait rw;
```

第二步：加载内核到内存

```
Phytium-Pi# ext4load mmc 0:1 0x90100000 boot/Image;
```

这时，串口会打印如下信息提示加载成功

```
28692992 bytes read in 6293 ms (4.3 MiB/s)
```

第三步：加载设备树到内存

```
Phytium-Pi# ext4load mmc 0:1 0x90000000 boot/phytiumpi_firefly.dtb;
```

这时，串口会打印如下信息提示加载成功

```
25125 bytes read in 13 ms (1.8 MiB/s)
```

第四步：引导启动内核

```
Phytium-Pi# booti 0x90100000 - 0x90000000;
```

打印信息如下

```
Moving Image from 0x90100000 to 0x90200000, end=91df0000
## Flattened Device Tree blob at 90000000
   Booting using the fdt blob at 0x90000000
   Loading Device Tree to 00000000f9c32000, end 00000000f9c3b224 ... OK
run in ft_board_setup
fdt_addr 00000000f9c32000
N: Phytium System Service Call: 0xc2000005
mb_count = 0x1
```

```
mb_blocks[0].mb_size = 0x7c000000
mb_blocks[1].mb_size = 0xffa16920
fdt : remove memory@1fdt : dram size 0x17fa16920 update successfully

Starting kernel ...
/ {
        compatible = "phytium,pe2204";
        interrupt-parent = <0x00000001>;
        #address-cells = <0x00000002>;
… …                    // 打印信息太多，此处省略
[      0.000000] Booting Linux on physical CPU 0x0000000200 [0x700f3034]
[      0.000000] Linux version 5.10.209-rt101-phytium-embeded-v2.1
[      0.000000] Machine model: Phytium Pi Board
… …                              // 打印信息太多，此处省略
[   OK   ] Finished Hold until boot process finishes up.
[   OK   ] Finished Save/Restore Sound Card State.
[   OK   ] Started OpenBSD Secure Shell server.

Phytium Pi
phytiumpi login:              // Linux Shell 提示界面
```

这时，输入用户名 user 和密码 user 进入 Linux shell。

3.4 编译内核模块

关于如何编译内核外部模块，可参考 https://www.kernel.org/doc/html/latest/kbuild/modules.html

3.4.1 交叉编译内核模块

使用 Buildroot 的工具链来交叉编译内核模块，工具链位于 output/host/bin，工具链的 sysroot 为 output/host/aarch64-buildroot-linux-gnu/sysroot。

交叉编译内核外部模块的命令为：

```
$ make ARCH=arm64 \
CROSS_COMPILE=/home/xxx/phytium-pi-os/output/host/bin/aarch64-none-linux-gnu- \
-C /home/xxx/phytium-pi-os/output/build/linux-xxx/ \
M=$PWD \
modules
```

3.4.2 飞腾派开发板上编译内核模块

利用 linux-headers 可以在开发板上进行内核模块编译，软链接/lib/modules/version/build 指向

linux-headers。

在开发板上编译内核外部模块的命令为：

```
make -C /lib/modules/xxx/build M=$PWD modules
```

3.5 Buildroot 编译新的应用软件

本节简单介绍如何通过 Buildroot 交叉编译能运行在开发板上的应用软件，完整的教程请参考 https://buildroot.org/downloads/manual/manual.html。

3.5.1 Buildroot 软件包介绍

Buildroot 中所有用户态的软件包都在 package 目录，每个软件包有自己的目录 package/<pkg>，其中<pkg>是小写的软件包名。这个目录包含：

（1）Config.in 文件，用 Kconfig 语言编写，描述了包的配置选项。

（2）<pkg>.mk 文件，用 make 编写，描述了包如何构建，即从哪里获取源码，如何编译和安装等。

（3）<pkg>.hash 文件，提供 hash 值，检查下载文件的完整性，如检查下载的软件包源码是否完整，这个文件是可选的。

（4）*.patch 文件，在编译之前应用于源码的补丁文件，这个文件是可选的。

（5）可能对包有用的其他文件。

3.5.2 编写 Buildroot 软件包

首先创建软件包的目录 package/<pkg>，然后编写该软件包中的文件。

Buildroot 中的软件包基本上由 Config.in 和<pkg>.mk 两个文件组成。关于如何编写这两个文件，大家可以参考 Buildroot 用户手册，这里简单概括一下。

（1）Config.in 文件中必须包含启用或禁用该包的选项，而且必须命名为 BR2_PACKAGE_<PKG>，其中<PKG>是大写的软件包名，这个选项的值是布尔类型。也可以定义其他功能选项来进一步配置该软件包。然后还必须在 package/Config.in 文件中包含该文件：

```
source "package/<pkg>/Config.in"
```

（2）<pkg>.mk 文件看起来不像普通的 Makefile 文件，而是一连串的变量定义，而且必须以大写的包名作为变量的前缀。最后以调用软件包的基础结构（package infrastructure）结束。变量告诉软件包的基础结构要做什么。

对于使用手写 Makefile 来编译的软件源码，在<pkg>.mk 中调用 generic-package 基础结构。generic-package 基础结构实现了包的下载、提取、打补丁。而配置、编译和安装由<pkg>.mk 文件描述。<pkg>.mk 文件中可以设置的变量及其含义，请参考 Buildroot 用户手册。

3.5.3 编译软件包

（1）单独编译软件包

```
$ cd xxx/phytiumpios
$ make <pkg>
```

编译结果在 output/build/<pkg>-<version>

（2）将软件包编译进根文件系统

在 phytiumpi_xxx_defconfig 中添加一行 BR2_PACKAGE_<PKG>=y

```
$ make phytiumpi_xxx_defconfig
$ make
```

A 更新记录

发布日期	版本	说明
2023-08-09	v1.0	初稿，介绍飞腾派 OS 的使用和开发
2024-02-04	v1.1	增加 OP-TEE 编译、增加树外构建和修改内核相关的 Buildroot 使用技巧
2024-05-08	v1.2	增加 3.1.7 节“OpenAMP 编译”和 3.2.3 节“减少二次编译时间”，修改 3.4 节 “编译内核模块”

B 扩展资料

https://gitee.com/phytium_embedded/phytium-embedded-docs