# 嵌入式 Linux 构建指南

尚硅谷

## 一　环境准备

本次实验宿主机使用全新的 Ubuntu20.04 环境，并在完成实验后用户目录下有以下结构的文件夹。

首先需要创建架构。

```
v3s-workspace
- linux
- partitions
-- boot
--- boot.scr
--- sun8i-v3s-licheepi-zero-dock.dtb
--- zImage
-- root
- u-boot
- modules
- buildroot
- boot.cmd
```

然后下载所有本次实验会使用到的代码和依赖。

```
# 安装依赖
sudo apt-get install flex bison gcc make gcc-arm-linux-gnueabihf
libncurses-dev swig python-dev device-tree-compiler python3-setuptools
python3-dev libssl-dev u-boot-tools g++ patch

# 下载 Mainline Linux,你可以在 https://www.kernel.org 寻找最新的 LTS 版本
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.19.tar.xz
# extract 是 zsh 提供的自动解压指令，可以替换成 tar 的对应格式解压指令
extract linux-6.1.19.tar.xz
mv linux-6.1.19 linux

# 下载 U-Boot，我们这里没有使用 LTS 版本，你可以进入 cd 进去后切换到 LTS 分支
git clone git://git.denx.de/u-boot.git

# 下载 Buildroot，你可以在 https://buildroot.org/downloads 寻找最新的版本
wget https://buildroot.org/downloads/buildroot-2023.02.tar.xz
extract buildroot-2023.02.tar.xz
mv buildroot-2023.02 buildroot
```

每一章节结束后请返回 v3s-workspace 目录。

# 二　编译 U-Boot

```
cd u-boot
# 使用荔枝派 Nano 的默认配置
make CROSS_COMPILE=arm-linux-gnueabihf- LicheePi_Zero_defconfig
# 编译
make CROSS_COMPILE=arm-linux-gnueabihf-
# 拷贝 U-Boot 镜像
cp u-boot-sunxi-with-spl.bin ../partitions
```

接下来我们需要准备 U-Boot 启动所需的配置文件，将以下内容写入 boot.cmd。

```
setenv bootargs console=tty0 console=ttyS0,115200 panic=5 rootwait
root=/dev/mmcblk0p2 rw
load mmc 0:1 0x43000000 sun8i-v3s-licheepi-zero-dock.dtb
load mmc 0:1 0x42000000 zImage
bootz 0x42000000 - 0x43000000
```

接着编译配置文件。

```
mkimage -C none -A arm -T script -d boot.cmd ./partitions/boot/boot.scr
```

# 三　编译 Linux Kernel

```
cd linux
# 使用 linux-sunxi 项目的默认配置，该项目主要包含全志各芯片的硬件支持文档和手册
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- sunxi_defconfig
# 进入内核配置菜单
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

在内核配置菜单中，将 Networking support > Wireless 中的选项全部选中。

我们需要修改 arch/arm/boot/dts/sun8i-v3s-licheepi-zero.dts 以启用以太网和 USB 支持。

```
/dts-v1/;
#include "sun8i-v3s.dtsi"
#include "sunxi-common-regulators.dtsi"
```

```
/ {
    model = "Lichee Pi Zero";
    compatible = "licheepi,licheepi-zero", "allwinner,sun8i-v3s";

    aliases {
        serial0 = &uart0;
        ethernet0 = &emac; /* 添加这一行 */
    };

    chosen {
        stdout-path = "serial0:115200n8";
    };

    leds {
        compatible = "gpio-leds";

        blue_led {
            label = "licheepi:blue:usr";
            gpios = <&pio 6 1 GPIO_ACTIVE_LOW>; /* PG1 */
        };

        green_led {
            label = "licheepi:green:usr";
            gpios = <&pio 6 0 GPIO_ACTIVE_LOW>; /* PG0 */
            default-state = "on";
        };

        red_led {
            label = "licheepi:red:usr";
            gpios = <&pio 6 2 GPIO_ACTIVE_LOW>; /* PG2 */
        };
    };

  /* 添加以下 soc 部分 */
    soc {
        ehci0: usb@01c1a000 {
            compatible = "allwinner,sun8i-v3s-ehci", "generic-ehci";
            reg = <0x01c1a000 0x100>;
            interrupts = <GIC_SPI 72 IRQ_TYPE_LEVEL_HIGH>;
            clocks = <&ccu CLK_BUS_EHCI0>, <&ccu CLK_BUS_OHCI0>;
            resets = <&ccu RST_BUS_EHCI0>, <&ccu RST_BUS_OHCI0>;
            status = "okay";
        };

        ohci0: usb@01c1a400 {
            compatible = "allwinner,sun8i-v3s-ohci", "generic-ohci";
            reg = <0x01c1a400 0x100>;
            interrupts = <GIC_SPI 73 IRQ_TYPE_LEVEL_HIGH>;
```

```
            clocks = <&ccu CLK_BUS_EHCI0>, <&ccu CLK_BUS_OHCI0>,
            <&ccu CLK_USB_OHCI0>;
            resets = <&ccu RST_BUS_EHCI0>, <&ccu RST_BUS_OHCI0>;
            status = "okay";
        };
    };
};

&mmc0 {
    broken-cd;
    bus-width = <4>;
    vmmc-supply = <&reg_vcc3v3>;
    status = "okay";
};

&uart0 {
    pinctrl-0 = <&uart0_pb_pins>;
    pinctrl-names = "default";
    status = "okay";
};

&usb_otg {
    dr_mode = "host";
    status = "okay";
};

&usbphy {
    usb0_id_det-gpios = <&pio 5 6 GPIO_ACTIVE_LOW>;
    status = "okay";
};

/* 添加 emac 部分 */
&emac {
    allwinner,leds-active-low;
    status = "okay";
};
```

接着执行以下指令。

```
# 编译内核，-j4 的 4 可以修改为你的 CPU 核心数
ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make -j4 zImage
# 编译 DTB 文件，本文件用于 Kernel 识别外设，是 Mainline Kernel 不可缺少的部分，-j4
的 4 可以修改为你的 CPU 核心数
ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make -j4 dtbs
# 编译 Modules，-j4 的 4 可以修改为你的 CPU 核心数
ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make -j4 modules
# 安装模块
```

```
ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- INSTALL_MOD_PATH=../modules
make modules modules_install
# 拷贝生成的 zImage 内核镜像和 DTB 文件
cp arch/arm/boot/zImage ../partitions/boot
cp arch/arm/boot/dts/sun8i-v3s-licheepi-zero-dock.dtb ../partitions/boot
```

# 四　编译 Buildroot

我们使用 Buildroot 默认的 BusyBox 程序和 GLIBC，如果需要剪裁大小，可以选择其他的 C 支持库。

```
cd buildroot
# 进入配置菜单
make menuconfig
```

| 配置位置 | 操作 | 用途 |
| --- | --- | --- |
| Target options | Target Arch 设置为 ARM (little endian) | 设置大小端 |
| Target options | Target Arch Variant 设置为 Cortex-A7 | 设置 CPU 架构 |
| Toolchain | Kernel Headers 设置为你下载的 LTS 版本内核对应的版本号 | 匹配内核版本 |
| Target packages > Networking applications | hostapd | 开启 |
| Target packages > Networking applications | hostapd > Enable hostap driver | 开启 |
| Target packages > Networking applications | hostapd > Enable nl80211 driver | 开启 |
| Target packages > Networking applications | hostapd > Enable ACS | 开启 |
| Target packages > Networking applications | hostapd > Enable EAP | 开启 |
| Target packages > Networking applications | hostapd > Enable WPS | 开启 |

表 1　Buildroot 相关配置

| 配置位置 | 操作 | 用途 |
| --- | --- | --- |
| Target packages > Networking applications | openssh | 开启 |
| Target packages > Networking applications | openssh > client | 开启 |
| Target packages > Networking applications | openssh > server | 开启 |
| Target packages > Networking applications | openssh > key utilities | 开启 |
| Target packages > Networking applications | openssh > use sandboxing | 开启 |
| Target packages > Networking applications | wireless tools | 开启 |
| Target packages > Networking applications | wireless tools > Install shared library | 开启 |
| Target packages > Networking applications | wpa_supplicant | 开启 |
| Target packages > Libraries > Crypto | openssl support | openssl support 的子菜单部件全部选中 |

表 2　Buildroot 相关配置

同时修改 `partitions/root/etc/fstab` 文件中的 ext2 为 ext4。

# 五　写入 SD 卡

我的 SD 卡路径是`/dev/sdb`，可以通过 `sudo fdisk -l` 查看 SD 卡的路径。

```
cd partitions
# 清空分区表
sudo dd if=/dev/zero of=/dev/sdb bs=1M count=1
# 写入 U-Boot
sudo dd if=u-boot-sunxi-with-spl.bin of=/dev/sdb bs=1024 seek=8
```

```
# 写入分区表，请复制除了本行内的内容并执行
sudo blockdev --rereadpt /dev/sdb
cat <<EOT | sudo sfdisk /dev/sdb
1M,16M,c
,,L
EOT
```

```
# 格式化
sudo mkfs.vfat /dev/sdb1
sudo mkfs.ext4 /dev/sdb2
# 拷贝 boot 进入第一个分区
sudo mount /dev/sdb1 /mnt
sudo cp -R boot/* /mnt
sync
sudo umount /mnt
# 拷贝 rootfs 进入第二个分区
sudo mount /dev/sdb2 /mnt
sudo cp -R root/* /mnt
sync
sudo umount /mnt
```

# 六　将嵌入式 Linux 作为 ssh 服务器使用

连接串口，波特率设置为 115200 。

使用 putty 软件连接嵌入式 Linux 开发板。

```
vi /etc/ssh/sshd_config
# 将文件中的 PermitRootLogin 设置为 yes
# PermitRootLogin yes
# 保存退出

# 使用 passwd 设置密码
passwd
# 启动 sshd 服务
/usr/sbin/sshd
```

注：这里建议重启一下板子，否则 /etc/ssh/sshd_config 不一定能够生效，我实测时就是需要重启才可以

```
# 启动以太网服务
ifconfig eth0 up
udhcpc
```

# 七　安装无线网卡驱动

```
cd v3s-workspace
git clone https://github.com/al177/esp8089.git
cd esp8089
ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make -C ../linux/ M=${PWD}
modules

# 172.17.0.58 替换为你的电脑看到的 Linux 开发板的 ip 地址。
scp esp8089.ko root@172.17.0.58:/root/
ssh root@172.17.0.58
# 进入嵌入式 Linux 开发板之后的操作
# 加载 wifi 驱动
atguigu-pi>insmod esp8089.ko
atguigu-pi>ifconfig wlan0 up
atguigu-pi>wpa_passphrase your_SSID your_passphrase > your_SSID.conf
atguigu-pi>wpa_supplicant -B -i wlan0 -c your_SSID.conf
```