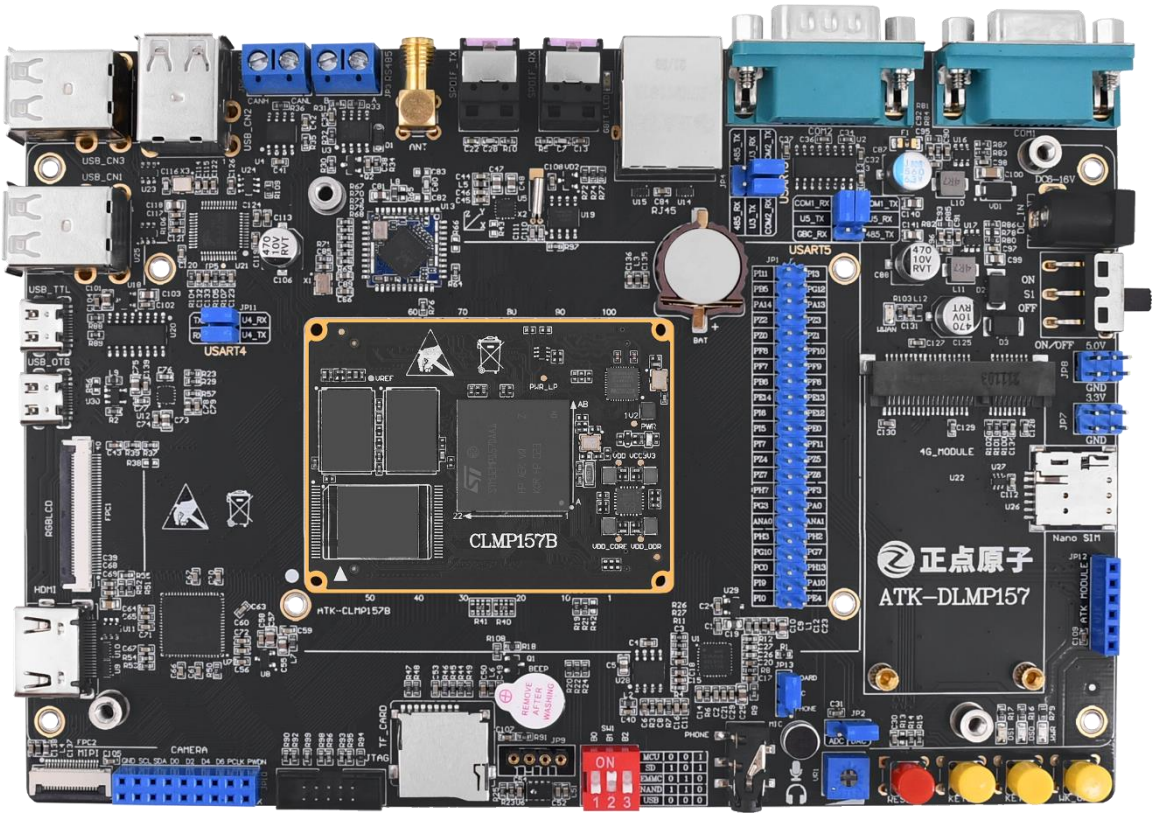
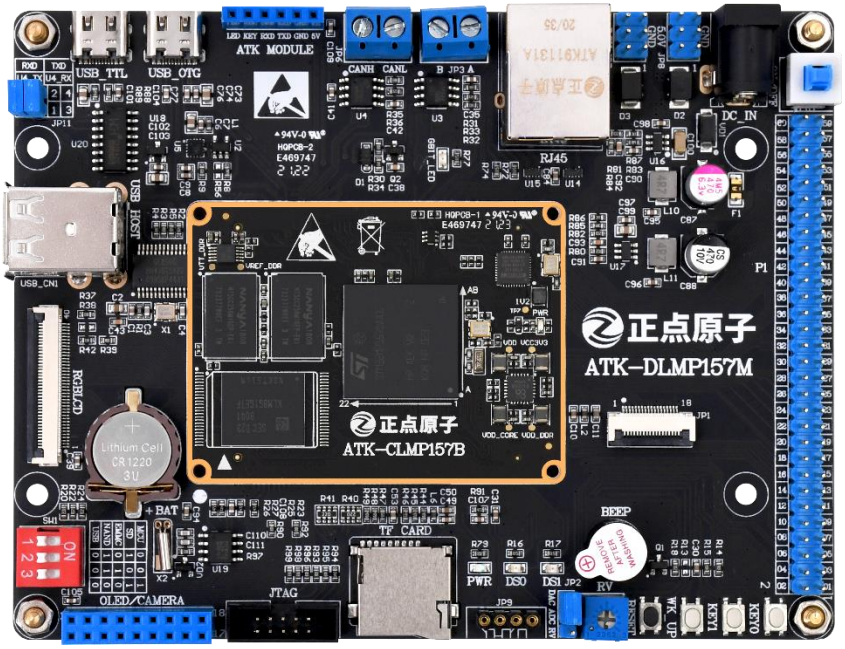
【正点原子】STM32MP157 Buildroot 构建根文件系统 V1.0





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

电话：020-38271790 传真：020-36773971

请关注正点原子公众号，资料发布更新我们会通知。
请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

扫码关注正点原子公众号

扫码下载“原子哥”APP

文档更新说明

版本	版本更新说明	负责人	校审	发布日期
V1.0	初稿：	正点原子 Linux 团队	正点原子 Linux 团队	2022.07.28

目录

前言

我们在前面有一份文档【正点原子】I.MX6U 移植 Qt5.12.9 V1.1.pdf 已经移植过 Qt 了（mp157 可以参考），方法十分麻烦，依赖 Busybox 根文件系统，还需要单独一个个的移植第三方库，甚至有些第三方库都不知道编译方法等等困难，所以编译出来的 Qt 库比较少，只适合那种只需要 Qt 支持功能较少的用户，比如某些用户就是只需要 Qt UI 显示和触摸即可，那么移植 Qt 部署到 Busybox 适合。实际上使用 Buildroot 构建 Qt 根文件系统，会比较省事，而且支持的库比较完整，也不用一个个去移植第三方库。使用 Buildroot 就能很方便构建一整套系统，包括 uboot 和内核。但是我们的正点原子 uboot 内核都是单独按驱动指南教程文档编译了，不在 Buildroot 里编译，所以编译 uboot 和内核不会在 Buildroot 里提及，也不是我们的重点。本篇文档重点是讲在 Buildroot 里如何编译 Qt，及使用 Qt，不会包含 Qt 库的支持全部功能，我们只是讲通用的方法编译 Qt，方便嵌入式系统移植 Qt 使用。如有错误联系本文档编写作者 QQ 1252699831 指正错误。

本文档所使用的环境：

- Windows 7 64bits，也适用于 Windows 8-10。不建议用 Windows 32 位来开发，Windows 32 位支持的内存大小有限，系统性能有限。
- Ubuntu16.04/Ubuntu18.04 64bits，其他 Ubuntu 版本未实测，但是方法一样。请自行测试。
- 要求读者会使用 FileZilla、WinSCP 及 Windows Git 进行 Ubuntu 与 Windows 间互传文件的方法。

第一章 Buildroot Qt 根文件系统构建

本章介绍 Buildroot 的学习方法，及 Buildroot 源码获取，编译配置 Qt 等。请读者提前熟悉 Buildroot。本文档是建立在读者会使用 Buildroot 的前提下，构建 Qt 根文件系统。

1.1 Buildroot 简介及使用方法

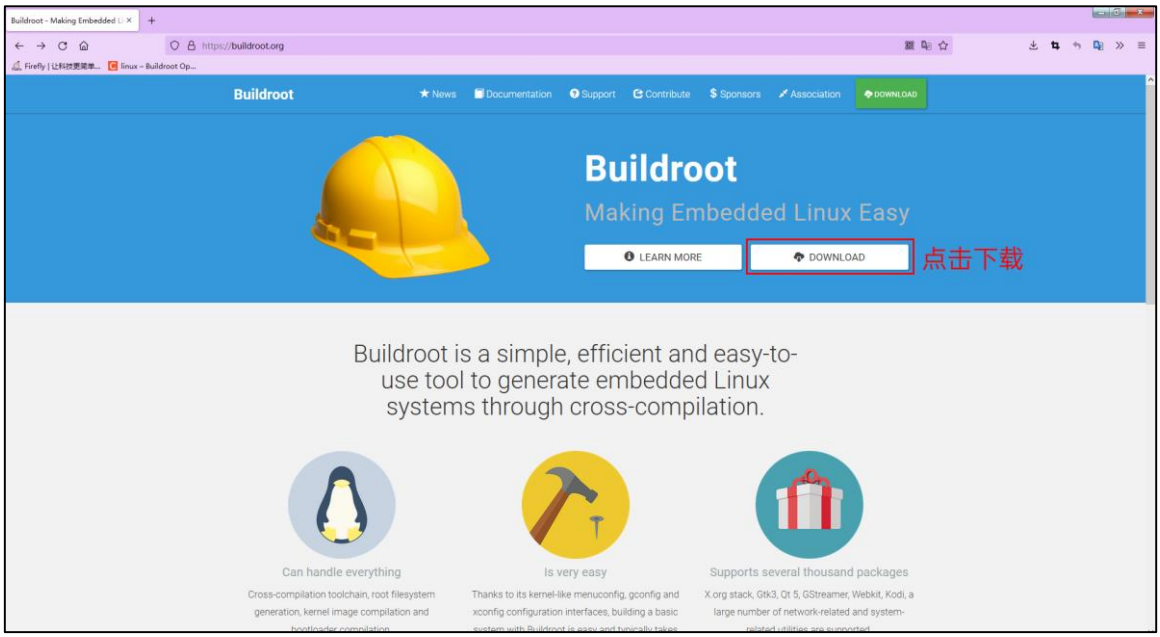
请 I.MX6ULL 用户先详看【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf 第 A1 章 Buildroot 根文件系统构建，MP157 用户【正点原子】STM32MP1 嵌入式 Linux 驱动开发指南 V2.x.pdf 第十九章 Buildroot 根文件系统构建学习使用 Buildroot 根文件系统，或者观看这两份文档对应的视频。本文档不介绍 Builroot 相关使用方法，只写操作步骤。请读者先充分熟悉 Buildroot 的使用。

1.2 获取 Buildroot 源码

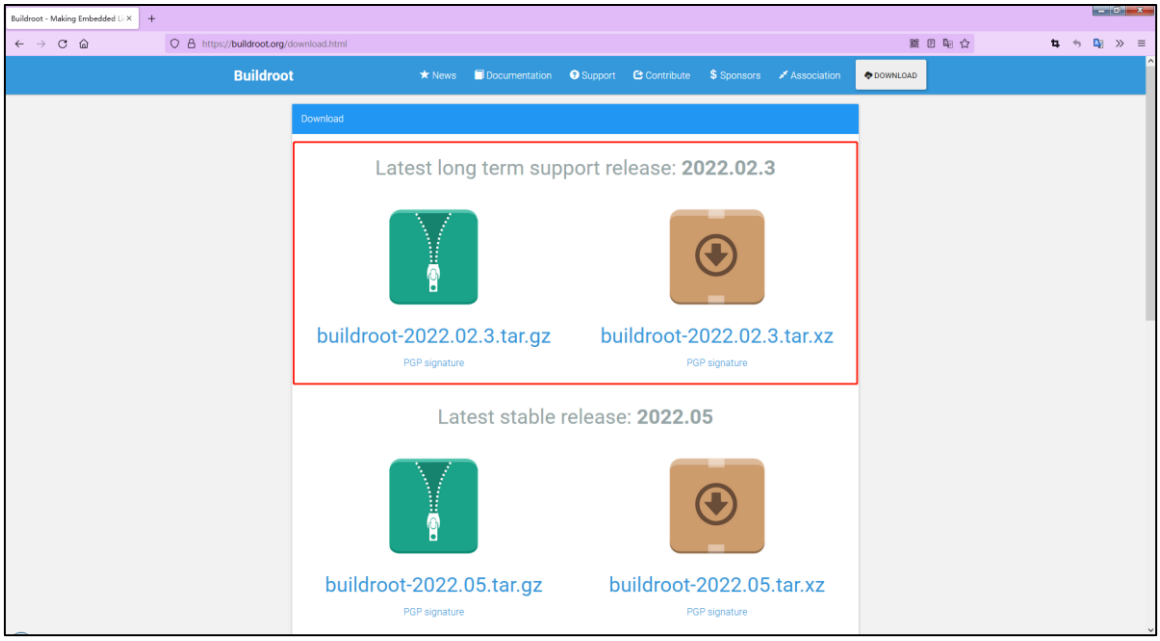
因为【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf 和【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf 使用的 Buildroot 版本不一样，我们也没有必要构建两个版本的 Qt Buildroot 根文件系统，请读者自行选择合适版本的 Buildroot 构建 Qt 根文件系统。

篇写这份教程的日期为为 2022.07.25，Buildroot 官网最新长期支持版本为 Buildroot-2022.02.3（LTS）。所以笔者使用最新的 Buildroot 作为测验。日后 Buildroot 版本会再升级，可能文档中的 Buildroot 版本过时，所以请读者根据自身情况酌情选择，版本不一定越新越好。

Buildroot 源码下载地址，https://buildroot.org/。打开官网页面如下。



下载页面如下，可以看到最新的版本已经为 Buildroot-2022.05，这个是稳定版本，也可以下载这个稳定版本。本次下载的是长期支持版本 Buildroot-2022.02.3。



直接点击上图红色框内的两个压缩包进行下载，xz 或者 gz 格式的都可以，注：浏览器下载需要比较长的时间，复制到迅雷会快很多。本文下载完成得到 buildroot-2022.02.3.tar.gz。然后将这个 buildroot-2022.02.3.tar.gz 压缩包拷贝到 Ubuntu 里。

本文拷贝到家目录。输入下面的指令进行解压。

```
tar xf buildroot-2022.02.3.tar.gz
```

解压完成如下

```
alientek@ubuntu:~$ tar xf buildroot-2022.02.3.tar.gz
alientek@ubuntu:~$ ls buildroot-2022.02.3
arch  board  boot  CHANGES  Config.in  Config.in.legacy  configs  COPYING  DEVEL
OPERS  docs  fs  linux  Makefile  Makefile.legacy  package  README  support  sys
tem  toolchain  utils
alientek@ubuntu:~$
```

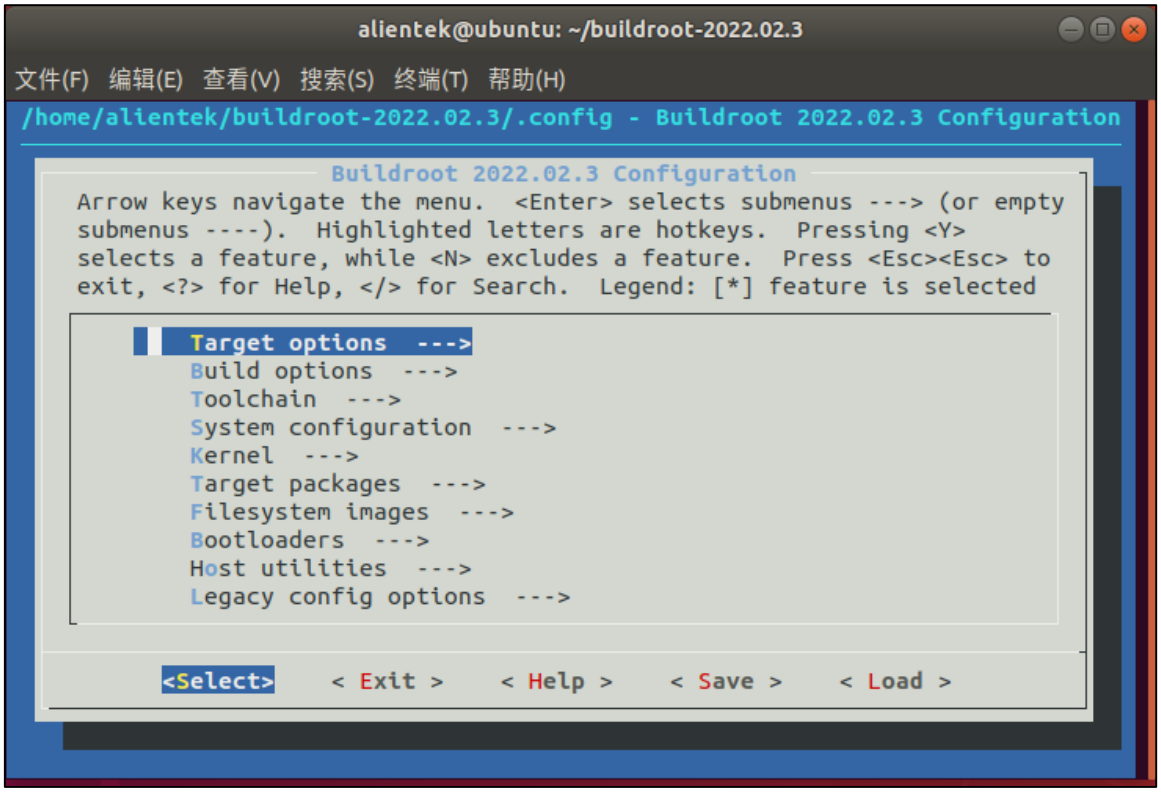
1.3 配置 Buildroot

进入 buildroot-2022.02.3 目录。输入下面的指令安装显示图形菜单需要的库，若已经安装则不需要装。

```
sudo apt-get install libncurses5-dev
```

打开配置菜单，输入下面的指令，如下图所示。

```
make menuconfig
```

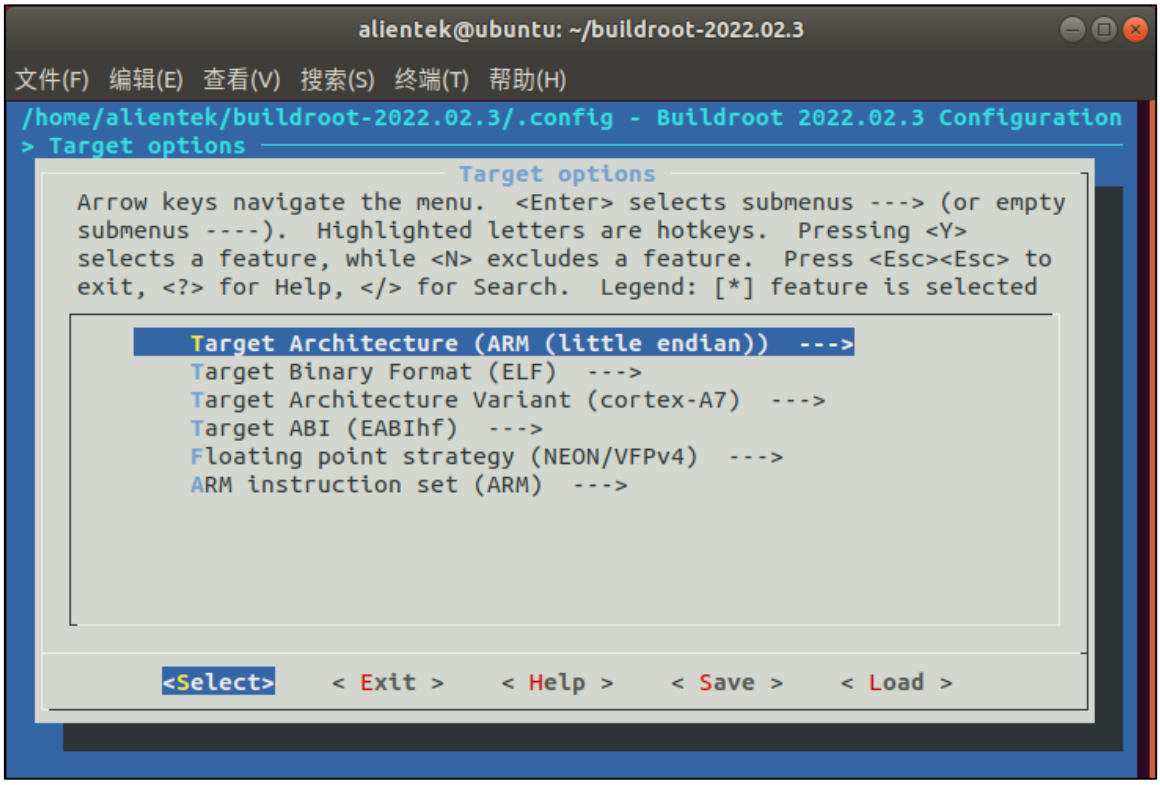


1.3.1 配置 Target options

（注若不想手动勾选，请直接看 1.4 小节配置好的文件）首先配置 Target options 选项，需要配置的项目和其对应的内容如下(“=”号后面是配置项要选择的内容！)：

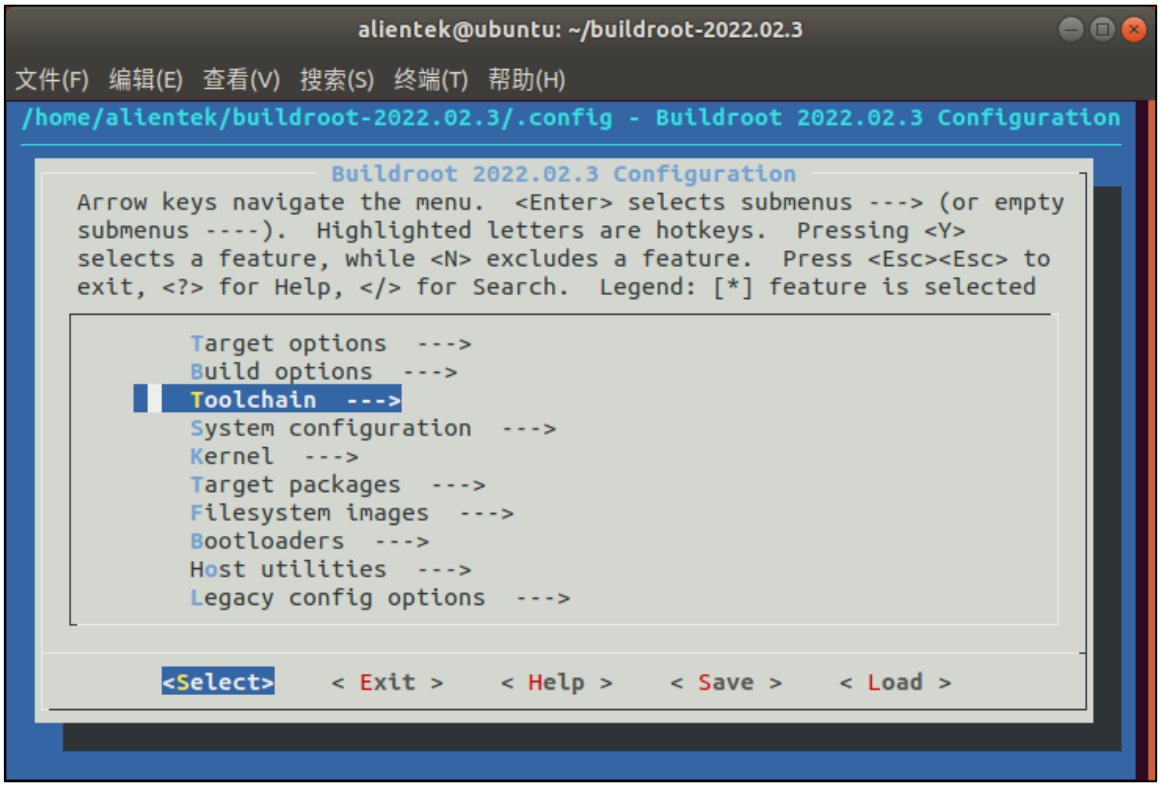
```
Target options
-> Target Architecture = ARM (little endian)
-> Target Binary Format = ELF
-> Target Architecture Variant = cortex-A7
-> Target ABI = EABIhf
-> Floating point strategy = NEON/VFPv4
-> ARM instruction set = ARM
```

配置完成如下图



1.3.2 配置交叉编译工具链

本文档不使用 Linux 驱动指南里的交叉编译器，内核 uboot 这些与根文件系统不使用同一个编译器也是没有关系的，所以为了省事，我们使用 Buildroot 里面的 ARM 编译器即可，这样也可以减少出错的概率。如下选项，我们勾选即可 。

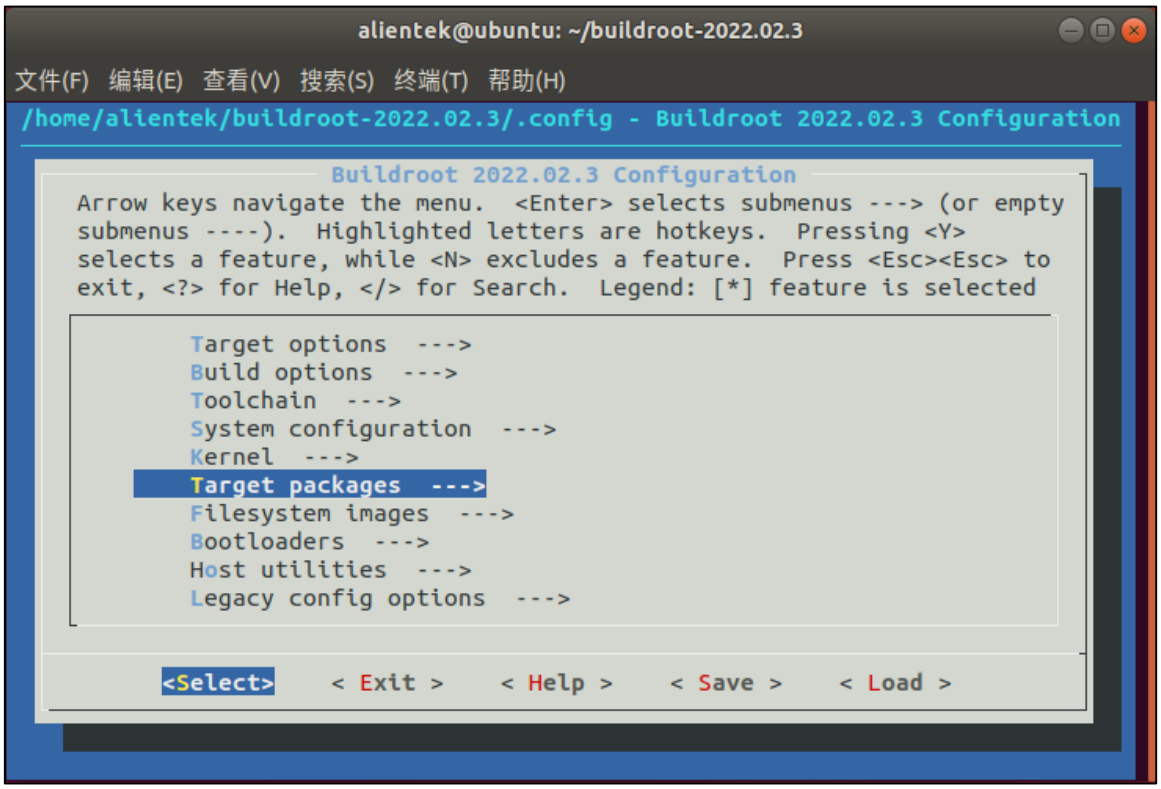


配置 Toolchain 选项，需要配置的项目和其对应的内容如下(“=”号后面是配置项要选择的内容！)：

```
Toolchain
    -> Toolchain type (Buildroot toolchain) = Buildroot toolchain
    -> GCC compiler Version (gcc 9.x)      = gcc 9.x
    -> [*] Enable toolchain locale/i18n support
    -> [*] Enable C++ support
    -> [*] Enable MMU support (NEW)
```

1.3.3 配置 Qt 编译选项

配置 Qt 选项，这里配置的选项有 Qt 的 gui 模块、Qt sql 模块、QtChart 模块、Quick 模块、Qt 虚拟键盘和 Qt 触摸插件 Tslib。可以看到像 tslib 插件这种第三方库可以在 Buildroot 里轻松配置，我们就不用单独的一个个移植第三方库了。为了减少编译时间，目前我们就配置如下，足够一些应对日常应用了。如需要支持更多模块功能可以自行勾选。




```
Target packages->
    ->Graphic libraries and applications (graphic/text)  --->
        -> [*] Qt5  --->
            -> [*]      Compile and install examples (with code)
            -> [*]      MySQL Plugin
            ->SQLite 3 support (Qt SQLite)      = Qt SQLite
            -> [*]        widgets module
            -> [*]        GIF support
            -> [*]    qt5charts
            -> [*]      Enable Tslib support
```

```
        -> [*]    qt5quickcontrols
        -> [*]    qt5quickcontrols2
        -> [*]    qt5virtualkeyboard
```

1.3.4 配置 System configuration

此选项用于设置一些系统配置，比如开发板名字、欢迎语、用户名、密码等。需要配置的项目和其对应的内容如下：

```
System configuration
    -> System hostname = atkbuildrootQtfs //平台名字，自行设置
    -> System banner = Welcome to alientek Qt rootfs //欢迎语
    -> Init system = BusyBox //使用 busybox
    -> /dev management = Dynamic using devtmpfs + mdev //使用 mdev
    -> [*] Enable root login with password (NEW) //使能登录密码
    -> Root password = root //登录密码为 root
```

在 System configuration 选项中可以配置平台名字，登录密码等信息。可以看出 buildroot 里面可以设置登录密码，但是作为实验，不建议大家设置登录密码，否则开发板每次重启都要输入密码，不方便开发。但是若要使用 ssh，设置密码是方便开发的。

1.3.5 配置 Filesystem images

此选项配置我们最终制作的根文件系统为什么格式的，配置如下：

```
Filesystem images
    -> [*] ext2/3/4 root filesystem //如果是 EMMC 或 SD 卡的话就用 ext3/ext4
    -> ext2/3/4 variant = ext4 //选择 ext4 格式
    -> exact size =1G //ext4 格式根文件系统 1GB(根据实际情况修改)
    -> [*] ubi image containing an ubifs root filesystem //如果使用 NAND 的话就用 u
bifs
```

可以看出，buildroot 可以直接制作出 ext4 格式的根文件系统，但是一般我们会自行往根文件系统里面添加很多其他的文件，所以产品开发完成以后需要自行打包根文件系统，然后烧写到开发板里面。不管是针对 EMMC 的 ext4 格式的根文件系统还是针对 NAND 的 ubi 格式的根文件系统，都要设置相应的大小，比如这里我们设置的 ex4 格式根文件系统大小为 1GB。

1.4 配置并构建 Buildroot 根文件系统

1.4.1 配置 Buildroot

在 Ubuntu 编辑为一个 atk_imx6ull_qt_defconfig 文件，然后拷贝到 Buildroot 源码的根目录 configs 下。

```
vi configs/atk_imx6ull_qt_defconfig
```

```
alientek@ubuntu:~/buildroot-2022.02.3$ vi configs/atk_imx6ull_qt_defconfig
alientek@ubuntu:~/buildroot-2022.02.3$
```

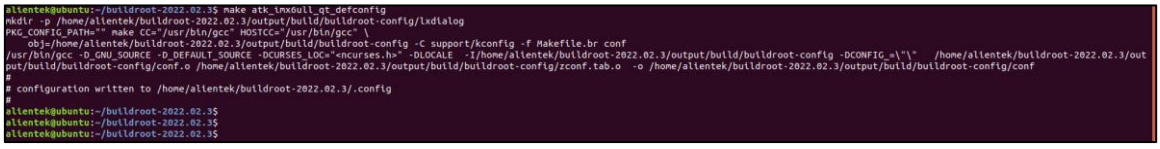
```
# ARM
```

```
BR2_arm=y
BR2_BINFMT_ELF=y
BR2_cortex_a7=y
BR2_ARM_EABIHF=y
BR2_ARM_FPU_NEON_VFPV4=y
BR2_ARM_INSTRUCTIONS_ARM=y

# toolchain
BR2_TOOLCHAIN_BUILDROOT=y
BR2_GCC_VERSION_9_X=y
BR2_TOOLCHAIN_BUILDROOT_CXX=y
BR2_USE_MMU=y
BR2_TOOLCHAIN_BUILDROOT_LOCALE=y

# Qt5
BR2_PACKAGE_QT5=y
BR2_PACKAGE_QT5BASE=y
BR2_PACKAGE_QT5BASE_EXAMPLES=y
BR2_PACKAGE_QT5BASE_MYSQL=y
BR2_PACKAGE_QT5BASE_SQLITE_QT=y
BR2_PACKAGE_QT5BASE_GUI=y
BR2_PACKAGE_QT5BASE_WIDGETS=y
BR2_PACKAGE_QT5BASE_LINUXFB=y
BR2_PACKAGE_QT5BASE_TSLIB=y
BR2_PACKAGE_QT5QUICKCONTROLS=y
BR2_PACKAGE_QT5QUICKCONTROLS2=y
BR2_PACKAGE_QT5VIRTUALKEYBOARD=y
BR2_PACKAGE_QT5BASE_GIF=y
BR2_PACKAGE_QT5BASE_JPEG=y
BR2_PACKAGE_QT5BASE_PNG=y
BR2_PACKAGE_QT5CHARTS=y

# System configuration
BR2_TARGET_GENERIC_HOSTNAME="atkbuildrootQtfs"
BR2_TARGET_GENERIC_ISSUE="Welcome to alientek buildroot Qt rootfs"
BR2_INIT_BUSYBOX=y
BR2_ROOTFS_DEVICE_CREATION_DYNAMIC_EUDEV=y

# filesystem / image
BR2_TARGET_ROOTFS_EXT4=y
BR2_TARGET_ROOTFS_EXT2_SIZE="1GB"
BR2_TARGET_GENERIC_ROOT_PASSWD="root"
```

```
#  - tiny ssh server and sftp server
BR2_PACKAGE_DROPBEAR=y
BR2_PACKAGE_GESFTPSERVER=y

BR2_TARGET_GENERIC_GETTY_PORT="ttySTM0"
```

输入 make atk_stm32mp157_qt_defconfig 开始配置

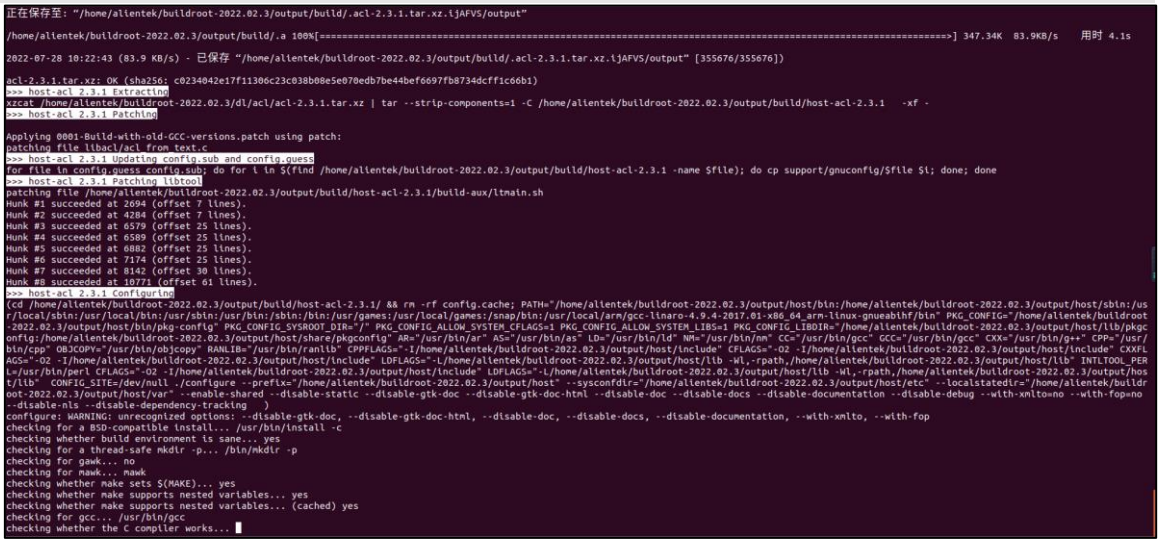
```
make atk_stm32mp157_qt_defconfig
```



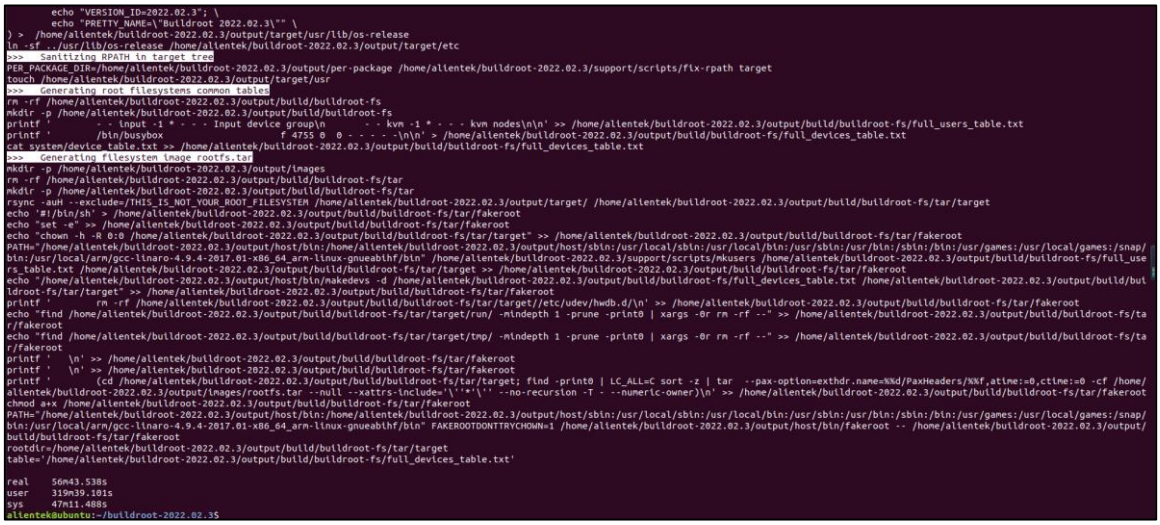
1.4.2 构建 Buildroot

输入 make -j x 进行构建， time 用于计算时间。x 参数根据个人 Ubuntu 配置取值。或者直接输入 make。

```
make -j 16
```

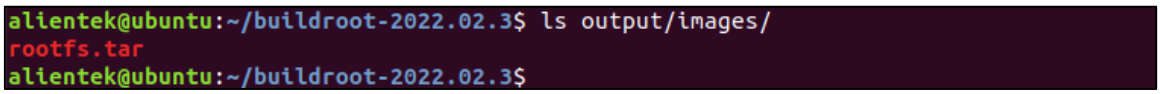


如上图，可以看到已经开始构建了，切记你的 Ubuntu 要联网！buildroot 编译的时候会先从网上下载所需的软件源码，有些软件源码可能下载不下来，这个时候就需要我们自行处理了。详细可以看驱动指南里的 Buildroot 部分。



构建成功如上图。中间暂停了一次，花了 80 分钟左右时间。

输入 ls output/images/查看生成根文件系统，如下图。



第二章 测试 Buildroot Qt 根文件系统

将生成的 Buildroot Qt 根文件系统烧写到开发板测试。

2.1 烧写 Buildroot Qt 根文件系统

将生成的 rootfs.tar 替换到 I.MX6U TF 卡系统启动卡的 rootfs 分区，先删除 rootfs 的根文件系统文件再解压过去，如果不会制作 TF 卡系统启动卡，请先看正点原子快速体验文档固件烧写章节，通过 nfs 挂载等方法测试生成的 Buildroot 根文件系统即可，不再累述烧写文件系统部分的内容。

可以看到解压到 TF 卡系统启动卡里已经安装了 Qt 库。

```
alientek@ubuntu:~/buildroot-2022.02.3$ ls /media/alientek/rootfs/usr/lib/
libblkid.so                libncurses.so              libQt5Concurrent.so.5.15.8    libQt5QmlWorkerScript.so.5      libQt5QuickWidgets.so.5.15.8  libreadline.so.8
libcurses.so               libncurses.so.6            libQt5Core.so                 libQt5QmlWorkerScript.so.5.15   libQt5Sql.so                  libreadline.so.8.1
libform.so                 libncurses.so.6.1          libQt5Core.so.5               libQt5QmlWorkerScript.so.5.15.8 libQt5Sql.so.5                libstdc++.so
libform.so.6               libpanel.so                libQt5Core.so.5.15            libQt5QuickControls2.so         libQt5Sql.so.5.15             libstdc++.so.6
libform.so.6.1             libpanel.so.6              libQt5Core.so.5.15.8          libQt5QuickControls2.so.5       libQt5Sql.so.5.15.8           libstdc++.so.6.0.28
libfreetype.so             libpanel.so.6.1            libQt5Gui.so                  libQt5QuickControls2.so.5.15    libQt5Svg.so                  libstdc++.so.6.0.28-gdb.py
libfreetype.so.6           libpcre2-16.so             libQt5Gui.so.5                libQt5QuickControls2.so.5.15.8  libQt5Svg.so.5                libts.so
libfreetype.so.6.18.1      libpcre2-16.so.0           libQt5Gui.so.5.15             libQt5QuickShapes.so            libQt5Svg.so.5.15             libts.so.0
libhistory.so              libpcre2-16.so.0.11.0      libQt5Gui.so.5.15.8           libQt5QuickShapes.so.5          libQt5Svg.so.5.15.8           libts.so.0.10.4
libhistory.so.8            libpcre2-8.so              libQt5Network.so              libQt5QuickShapes.so.5.15       libQt5Test.so                 libturbojpeg.so
libhistory.so.8.1          libpcre2-8.so.0            libQt5Network.so.5            libQt5QuickShapes.so.5.15.8     libQt5Test.so.5               libturbojpeg.so.0
libjpeg.so                 libpcre2-8.so.0.11.0       libQt5Network.so.5.15         libQt5Quick.so                  libQt5Test.so.5.15            libturbojpeg.so.0.2.0
libjpeg.so.8               libpcre2-posix.so          libQt5Network.so.5.15.8       libQt5Quick.so.5                libQt5Test.so.5.15.8          libudev.so
libjpeg.so.8.2.2           libpcre2-posix.so.3        libQt5PrintSupport.so         libQt5Quick.so.5.15             libQt5VirtualKeyboard.so      libz.so
libkmod.so                 libpcre2-posix.so.3.0.2    libQt5PrintSupport.so.5       libQt5Quick.so.5.15.8           libQt5VirtualKeyboard.so.5    libz.so.1
libkmod.so.2               libpng16.so                libQt5PrintSupport.so.5.15    libQt5QuickTemplates2.so        libQt5VirtualKeyboard.so.5.15 libz.so.1.2.12
libkmod.so.2.3.7           libpng16.so.16             libQt5PrintSupport.so.5.15.8  libQt5QuickTemplates2.so.5      libQt5VirtualKeyboard.so.5.15.8 metatypes
libmenu.so                 libpng16.so.16.37.0        libQt5QmlModels.so            libQt5QuickTemplates2.so.5.15   libQt5Widgets.so              os-release
libmenu.so.6               libpng.so                  libQt5QmlModels.so.5          libQt5QuickTemplates2.so.5.15.8 libQt5Widgets.so.5            qt
libmenu.so.6.1             libQt5Charts.so            libQt5QmlModels.so.5.15       libQt5QuickTest.so              libQt5Widgets.so.5.15         terminfo
libmysqlclient_r.so        libQt5Charts.so.5          libQt5QmlModels.so.5.15.8     libQt5QuickTest.so.5            libQt5Widgets.so.5.15.8       ts
libmysqlclient_r.so.16     libQt5Charts.so.5.15       libQt5Qml.so                  libQt5QuickTest.so.5.15         libQt5Xml.so
libmysqlclient_r.so.16.0.0 libQt5Charts.so.5.15.8     libQt5Qml.so.5                libQt5QuickTest.so.5.15.8       libQt5Xml.so.5
libmysqlclient.so          libQt5Concurrent.so        libQt5Qml.so.5.15             libQt5QuickWidgets.so           libQt5Xml.so.5.15
libmysqlclient.so.16       libQt5Concurrent.so.5      libQt5Qml.so.5.15.8           libQt5QuickWidgets.so.5         libQt5Xml.so.5.15.8
libmysqlclient.so.16.0.0   libQt5Concurrent.so.5.15   libQt5QmlWorkerScript.so      libQt5QuickWidgets.so.5.15      libreadline.so
alientek@ubuntu:~/buildroot-2022.02.3$
```

2.2 测试 Buildroot Qt 根文件系统启动

将 TF 卡插到开发板，拨码拨到 SD 卡启动方式，系统启动后如下。可以看到启动非常迅速，花费的时间比出厂系统的短，启动时间短可能功能就不会相对那么多，凡事都有两面性。

我们看到下面要求输入密码登录。配置的用户是 root，密码 root。输入两次 root 即可！

```
[    4.542278] cfg80211: failed to load regulatory.db
[    4.542725] ALSA device list:
[    4.549729]   #0: STM32MP1-DK
[    4.555900] Freeing unused kernel memory: 1024K
[    4.588019] Run /init as init process
Starting version 244.3+
[    7.292721] EXT4-fs (mmcblk1p5): recovery complete
[    7.300731] EXT4-fs (mmcblk1p5): mounted filesystem with ordered data mode. Opts: (null)
RESIZE ROOTFS [/dev/mmcblk1p5]
resize2fs 1.45.4 (23-Sep-2019)
Filesystem at /dev/mmcblk1p5 is mou[    7.381170] EXT4-fs (mmcblk1p5): resizing filesystem from 262144 to 3872503 blocks
nted on /rootfs; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 2
[    7.833932] EXT4-fs (mmcblk1p5): resized filesystem to 3872503
The filesystem on /dev/mmcblk1p5 is now 3872503 (4k) blocks long.

RESIZE BOOTFS [/dev/mmcblk1p4]
e2fsck 1.45.4 (23-Sep-2019)
bootfs: recovering journal
Checking for bad blocks (read-only test): done
bootfs: Updating bad block inode.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information

bootfs: ***** FILE SYSTEM WAS MODIFIED *****
bootfs: 725/16384 files (0.3% non-contiguous), 39000/65536 blocks
resize2fs 1.45.4 (23-Sep-2019)
Please run 'e2fsck -f /dev/mmcblk1p4' first.

/init: eval: line 1: resize_enabled: not found
mount: mounting proc on /proc failed: Device or resource busy
[   12.226816] EXT4-fs (mmcblk1p5): re-mounted. Opts: (null)
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Populating /dev using udev: [   12.494042] udevd[326]: starting version 3.2.11
[   12.538066] udevd[327]: starting eudev-3.2.11
done
Saving random seed: OK
Starting network: OK
Starting dropbear sshd: OK

Welcome to alientek buildroot Qt rootfs
atkbuildrootQtfs login:
```

登录后如下

```
Starting network: OK
Starting dropbear sshd: OK

Welcome to alientek buildroot Qt rootfs
atkbuildrootQtfs login: [  111.103879] random: nonblocking pool is initialized
root
Password:
# who
root           ttymxc0         00:00   Jul 21 09:09:25
#
```

2.3 测试 tslib 能否正常触摸

与 Qt 移植文档一样。设置好 tslib 的相关环境变量，先测试 tslib 能否正常触摸即可！由于 tslib 编译后相关的库都在/usr/lib/下，我们就不用设置它的环境变量了。

我们先输入 ts_test 指令测试能否正常画线，能正常画线就代表能触摸，也就不用再配置了。如果不能触摸可能是它默认指向的事件不对，需要调整。

```
ts_test
```

输入上面的指令后发现不需要配置 tslib 的环境变量也可以正常触摸。

若需要调整触摸事件，在/etc/profile 文件里加入以下内容。

```
export  TSLIB_TSDEVICE=/dev/input/event1      // eventX  X 可能取值 0  1  …
export  QT_QPA_FB_TSLIB=1                    // 若不能触摸，加入这个环境变量
```

2.4 配置 Qt 的运行环境变量

与 Qt 移植文档一样。设置好 Qt 的相关环境变量，先测试 tslib 能否正常触摸即可！由于 Qt 编译后相关的库都在/usr/lib/下，所以像配置 Qt 库的路径的环境变量也不需要再设置了。

在/etc/profile 文件里加入以下内容，指定 Qt 运行的平台插件即可。

```
vi  /etc/profile
```

插入 export QT_QPA_PLATFORM=linuxfb 环境变量。

```
export PATH="/bin:/sbin:/usr/bin:/usr/sbin"

if [ "$PS1" ]; then
        if [ "`id -u`" -eq 0 ]; then
                export PS1='# '
        else
                export PS1='$ '
        fi
fi

export EDITOR='/bin/vi'

# Source configuration files from /etc/profile.d
for i in /etc/profile.d/*.sh ; do
        if [ -r "$i" ]; then
                . $i
        fi
done
unset i
export QT_QPA_PLATFORM=linuxfb
```

如需要显示文字，就需要字库了。编辑/etc/profile 在末尾添加 export QT_QPA_FONTDIR=/usr/share/fonts。请自行将 windows 下的（路径 C:\Windows\Fonts）下的中文字库放到新建一个/usr/share/fonts/目录下就可以了。若例程有使用到字符，会显示找到不字库。注意 Windwos 的字库仅为个人学习使用，不要用于商业用途！很多都是有版权的！我们可以使用宋体，因为宋体免费。

使能 Qt 运行环境变量，下次开机就不用再 source 了，开机会运行 profile 里的环境变量。

```
source  /etc/profile
```

2.5 测试 Qt 能否正常运行

前面我们配置 Qt 的时候，把 Qt 的 examples，也就是它里面自带的一些例子，编译到我们的文件系统里了。我们就可以在我们的 Buildroot Qt 根文件系统直接运行 Qt 例子了。

编译好的 Qt 例子的路径为/usr/lib/qt/examples/。

那么我们就直接执行里面的例子程序。如下，随机一个执行程序，我们选择了如下路径的一个可执行程序来执行。

```
/usr/lib/qt/examples/widgets/animation/animatedtiles/animatedtiles
```

运行的界面如下，点击右下角的按钮，触摸也有反应。说明 Qt 的触摸也正常。

第三章 编译 Qt 项目

使用 Buildroot 里的交叉编译工具链编译 Qt 项目。我们需要知道，编译的 Qt 项目需要与编译 Buildroot Qt 根文件系统为同一个编译器。否则可能会报版本不对应之类的错误。

3.1 查看 Qt 版本

输入如下指令查看 Qt 版本。

output/host/usr/bin/qmake -v

```
alientek@ubuntu:~/buildroot-2022.02.3$ pwd
/home/alientek/buildroot-2022.02.3
alientek@ubuntu:~/buildroot-2022.02.3$ output/host/usr/bin/qmake -v
QMake version 3.1
Using Qt version 5.15.8 in /home/alientek/buildroot-2022.02.3/output/host/arm-buildroot-linux-uclibcgnueabihf/sysroot/usr/lib
alientek@ubuntu:~/buildroot-2022.02.3$
```

可以看到上面的结果为 Qt 5.15.8 版本。

3.2 命令行交叉编译 Qt 项目

与 Qt 移植文档一样，我们可以配置 Qt Creator 相关套件来编译 ARM 类型的 Qt 可执行程序，可以参考我们正点原子的 Qt 移植文档。这里我们就不浪费笔墨了。笔者喜欢用命令行编译 Qt 项目。

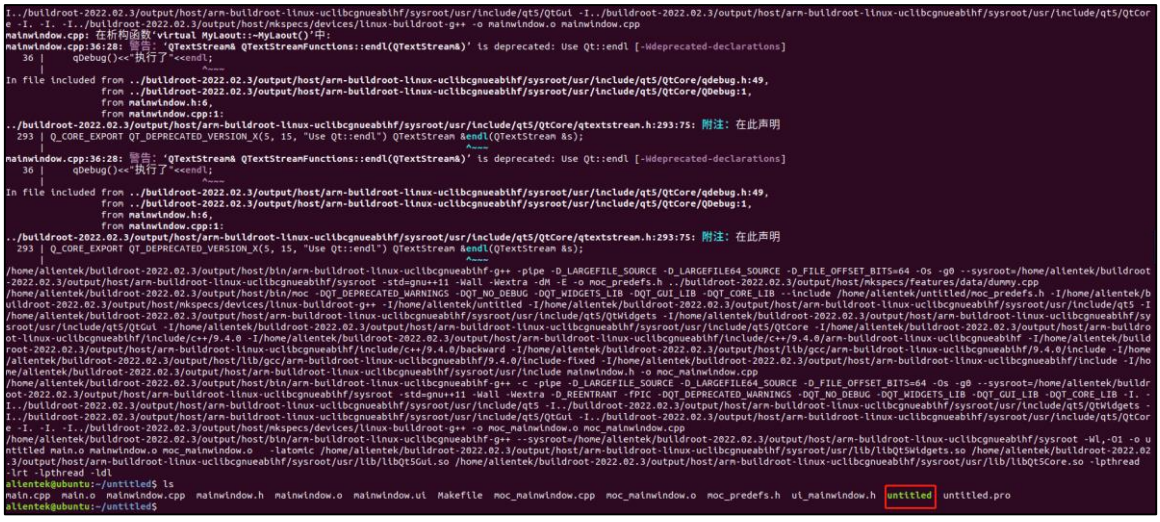
随意进入一个 Qt 项目路径下，路径下必须有 xx.pro 文件，如下面笔者家目录下有一个 untitled 的 Qt 项目。然后执行下面的指令。注意要改为自己的 Buildroot 的路径。

/home/alientek/buildroot-2022.02.3/output/host/usr/bin/qmake

```
alientek@ubuntu:~/untitled$ ls
main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui  untitled.pro
alientek@ubuntu:~/untitled$ /home/alientek/buildroot-2022.02.3/output/host/usr/bin/qmake
Info: creating stash file /home/alientek/untitled/.qmake.stash
alientek@ubuntu:~/untitled$ ls
main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui  Makefile  untitled.pro
alientek@ubuntu:~/untitled$
```

上面执行 qmake 后，会生成 Makefile 用于编译，我们就直接输入 make 编译即可！非常简单！

make



编译完成如上图，编译信息可能会报一些警告，这是因为编译器 g++版本与使用的 c++版本有差异，编译不报错即可！将生成的 untitled 拷贝到我们的 Buildroot Qt 根文件系统执行即可！

至此文档已结束，感谢大家支持正点原子！

附录-A