

ARK1668ED 搭建 Qt 交叉编译环境详细介绍

目录

ARK1668ED 搭建 Qt 交叉编译环境详细介绍	1
第一章 通过 Buildroot 下载编译 Qt5.15.11	2
1.1 配置 Buildroot 选中 Qt5	2
1.2 编译 BSP 源码	3
第二章 编译 BSP 生成交叉编译器	4
2.1 编译生成交叉编译器 SDK	4
2.2 解压交叉编译器 SDK	4
2.3 配置交叉编译器 SDK	4
第三章 配置 QCreator 的构建套件 Kit	5
3.1 下载 PC 版 Linux Qt	5
3.2 配置交叉编译构建套件	5
第四章 新建 Qt 测试工程	8
4.1 新建 Qt 工程	8
4.2 配置工程	8
4.3 添加一个测试界面	10
第五章 在 Arm 平台运行 Qt 编译生成的应用	12
5.1 拷贝测试文件到 USB	12
5.2 在开发板上运行测试文件	12
第六章 通过 Buildroot 裁剪 Qt 模块	13
6.1 重新配置 Buildroot 的 Qt 模块	13
6.2 重新编译 buildroot	14

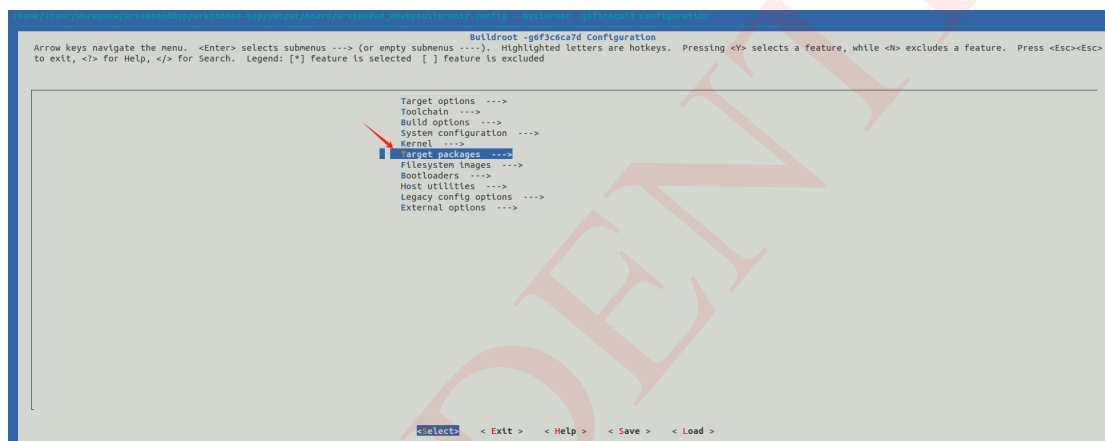
第一章 通过 Buildroot 下载编译 Qt5.15.11

1.1 配置 Buildroot 选中 Qt5

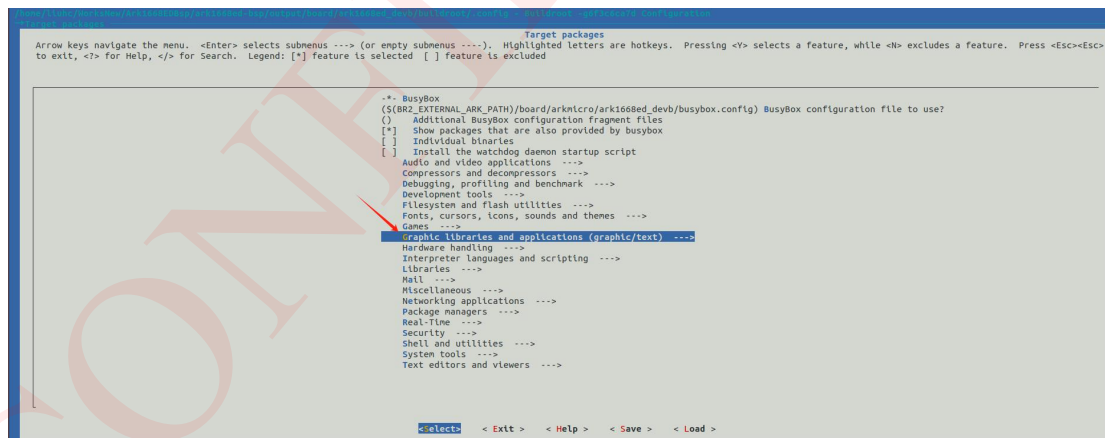
在 ark1668ed-bsp/build 下执行下图中的指令配置 Buildroot

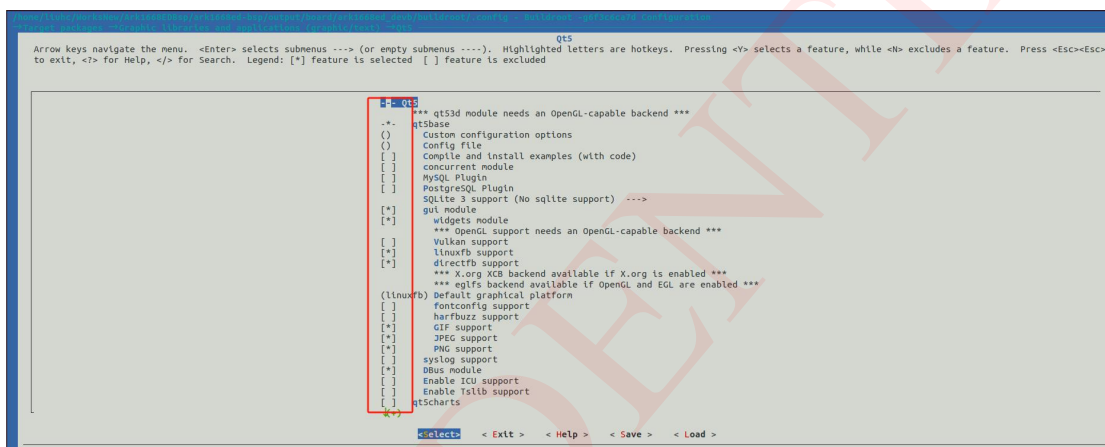
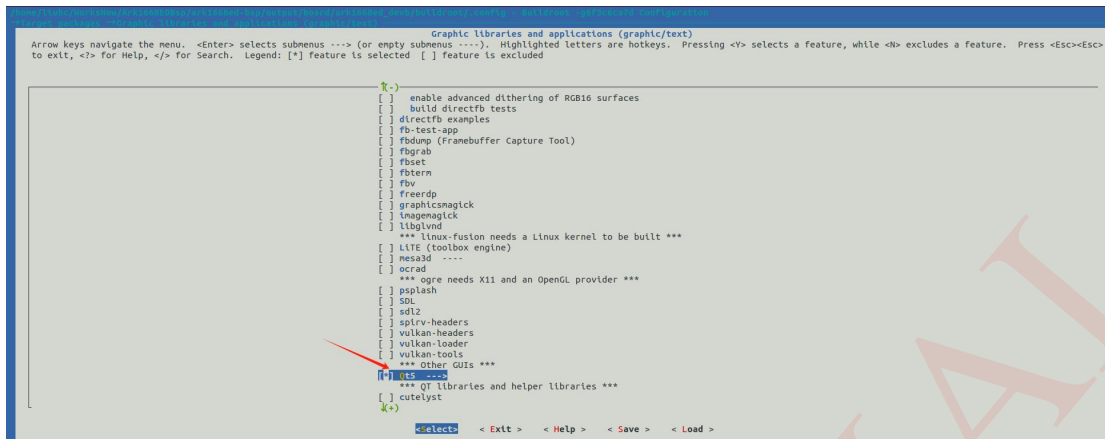
```
liu@liu@liu-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp$ cd build
liu@liu@liu-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
liu@liu@liu-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$ ./build.sh ark1668ed_devb config buildroot
config buildroot...
# GEN      /home/liu@liu@liu-virtual-machine/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/Makefile
#
# configuration written to /home/liu@liu@liu-virtual-machine/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/.config
#
# GEN      /home/liu@liu@liu-virtual-machine/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/Makefile
```

在界面中选择 Target Packages 选项



依次选择下图中的选项





1.2 编译 BSP 源码

此时，如果是第一次编译 BSP 就整体编译

[illegible]

如果已经执行过上图命令，就只需要单独编译 Buildroot

```
tiuhc@tiuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
tiuhc@tiuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
tiuhc@tiuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
tiuhc@tiuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$ ./build.sh ark1668ed_devb build buildroot^C
tiuhc@tiuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
tiuhc@tiuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
tiuhc@tiuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
tiuhc@tiuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
```

第二章 编译 BSP 生成交叉编译器

2.1 编译生成交叉编译器 SDK

在 BSP 源码（ark1668ed-bsp）编译后生成的 output 目录下编译生成 SDK。

```
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp$
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp$
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp$ ls
bootstrap build buildroot-2024.02.11 buildroot-external doc env.source linux luyuan-rgb.patch output tools u-boot
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp$ cd output/board/ark1668ed_devb/buildroot/
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot$
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot$
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot$ source ../../../../env.source
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot$
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot$ make sdk
>>> Finalizing host directory
>>> Finalizing target directory
mkdir -p /home/liuhc/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/host/etc/meson
```

2.2 解压交叉编译器 SDK

编译完成后，SDK 压缩文件（arm-buildroot-linux-gnueabihf_sdk-buildroot.tar.gz）在当前目录的 images 目录下。然后，拷贝生成的 SDK 压缩文件到虚拟机的任意目录下。这里选择拷贝到/home/liuhc/WorksNew/arkmicroNewSDK/目录。

```
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/images$ ls
arm-buildroot-linux-gnueabihf_sdk-buildroot.tar.gz rootfs.tar rootfs.ubi rootfs.ubifs
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/images$ cp arm-buildroot-linux-gnueabihf_sdk-buildroot.tar.gz /home/liuhc/WorksNew/arkmicroNewSDK/
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/images$ cd /home/liuhc/WorksNew/arkmicroNewSDK/
liuhc@liuhc-virtual-machine:~/WorksNew/arkmicroNewSDK$ ls
arm-buildroot-linux-gnueabihf_sdk-buildroot arm-buildroot-linux-gnueabihf_sdk-buildroot.tar.gz
liuhc@liuhc-virtual-machine:~/WorksNew/arkmicroNewSDK$ tar -xvf arm-buildroot-linux-gnueabihf_sdk-buildroot
```

2.3 配置交叉编译器 SDK

解压后生成目录（arm-buildroot-linux-gnueabihf_sdk-buildroot），然后在这个目录下执行脚本 relocate-sdk.sh

```
liuhc@liuhc-virtual-machine:~/WorksNew/arkmicroNewSDK/arm-buildroot-linux-gnueabihf_sdk-buildroot$ ls
arm-buildroot-linux-gnueabihf bin etc include lib lib64 libexec mkspecs opt relocate-sdk.shubin share usr
liuhc@liuhc-virtual-machine:~/WorksNew/arkmicroNewSDK/arm-buildroot-linux-gnueabihf_sdk-buildroot$ ./relocate-sdk.sh
Relocating the buildroot SDK from /home/liuhc/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/host to /home/liuhc/WorksNew/arkmicroNewSDK/arm-buildroot-linux-gnueabihf_sdk-buildroot ...
liuhc@liuhc-virtual-machine:~/WorksNew/arkmicroNewSDK/arm-buildroot-linux-gnueabihf_sdk-buildroot$
```

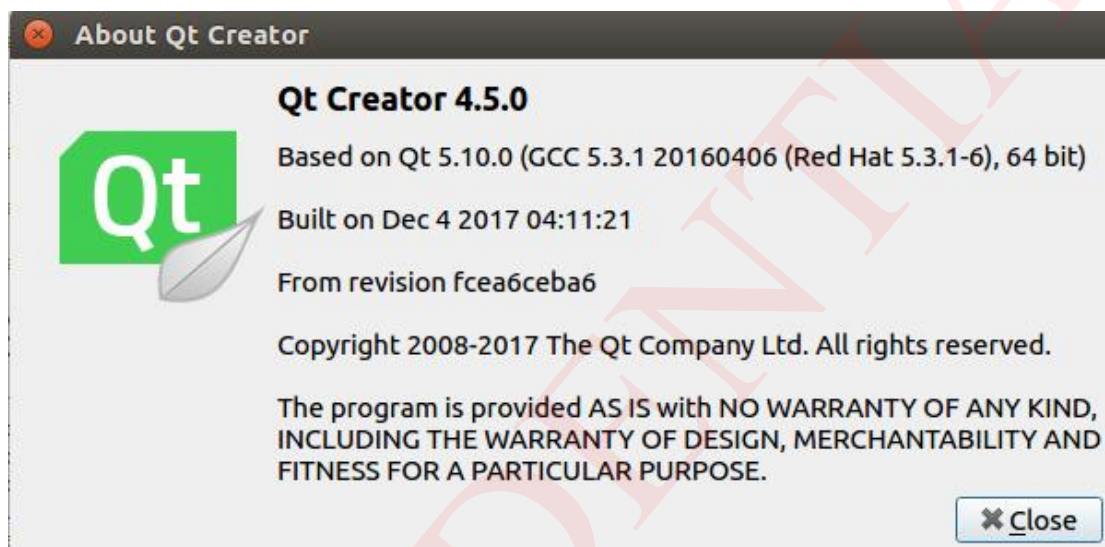
qmake、GCC 和 G++编译器在当前目录下的 bin 目录下，在此要留意这三个文件的路径，后面 QtCreater 配置构建套件会用到这 3 个工具。如下图：

```
liuhc@liuhc-virtual-machine:~/WorksNew/arkmicroNewSDK/arm-buildroot-linux-gnueabihf_sdk-buildroot/bin$ ls
qmake arm-none-linux-gnueabihf-gcc arm-none-linux-gnueabihf-g++
liuhc@liuhc-virtual-machine:~/WorksNew/arkmicroNewSDK/arm-buildroot-linux-gnueabihf_sdk-buildroot/bin$
```

第三章 配置 QtCreator 的构建套件 Kit

3.1 下载 PC 版 Linux Qt

从官网下载 pc 版本的 Linux Qt，只要是 Qt5 以上版本都可以，这里下载的是：
[qt-opensource-linux-x64-5.10.0.run](#)

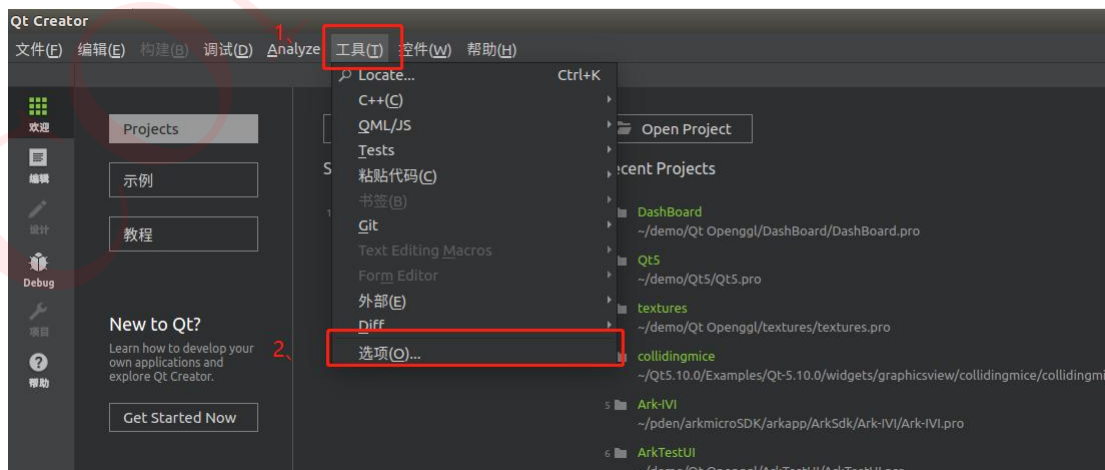


官方网址：http://download.qt.io/official_releases/qt/

下载完成后直接运行 `./qt-opensource-linux-x64-5.10.0.run` 全选傻瓜式安装即可！安装完成后接下来就是配置 qt 交叉编译环境了，如下：

3.2 配置交叉编译构建套件

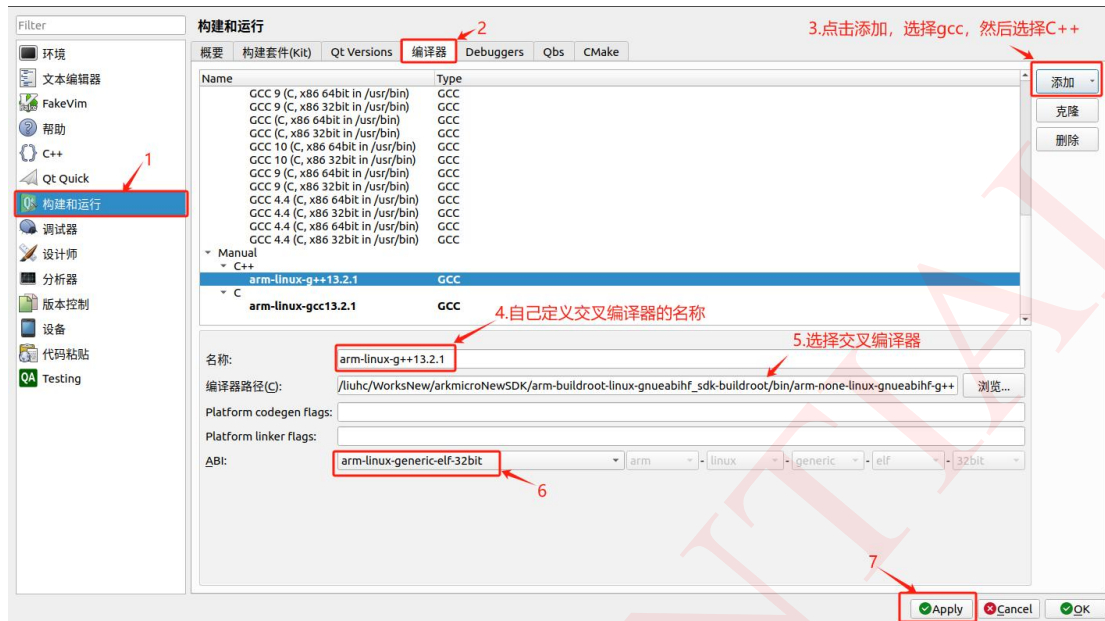
打开 QtCreator 的工具选项配置界面,步骤如下





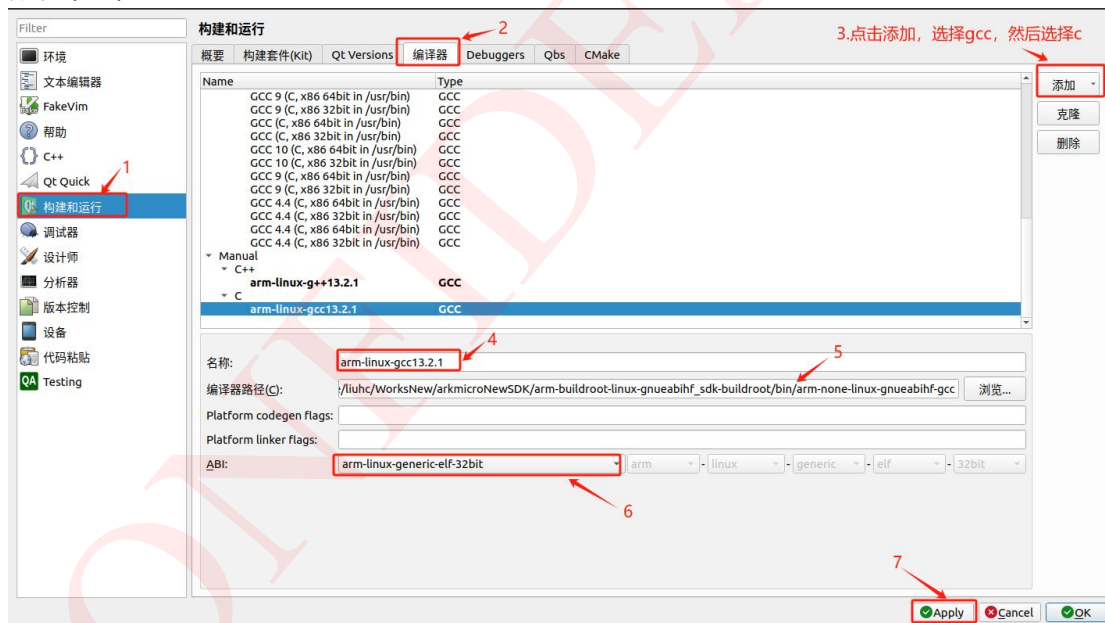
3.2.1 添加 G++交叉编译器

配置如下图：

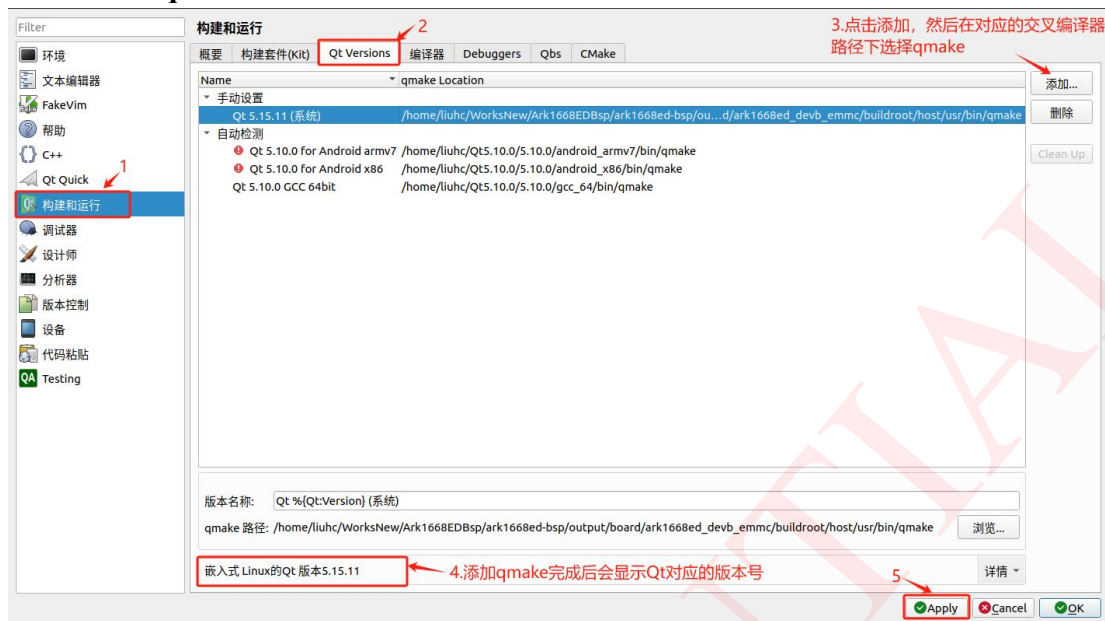


3.2.2 添加 GCC 交叉编译器

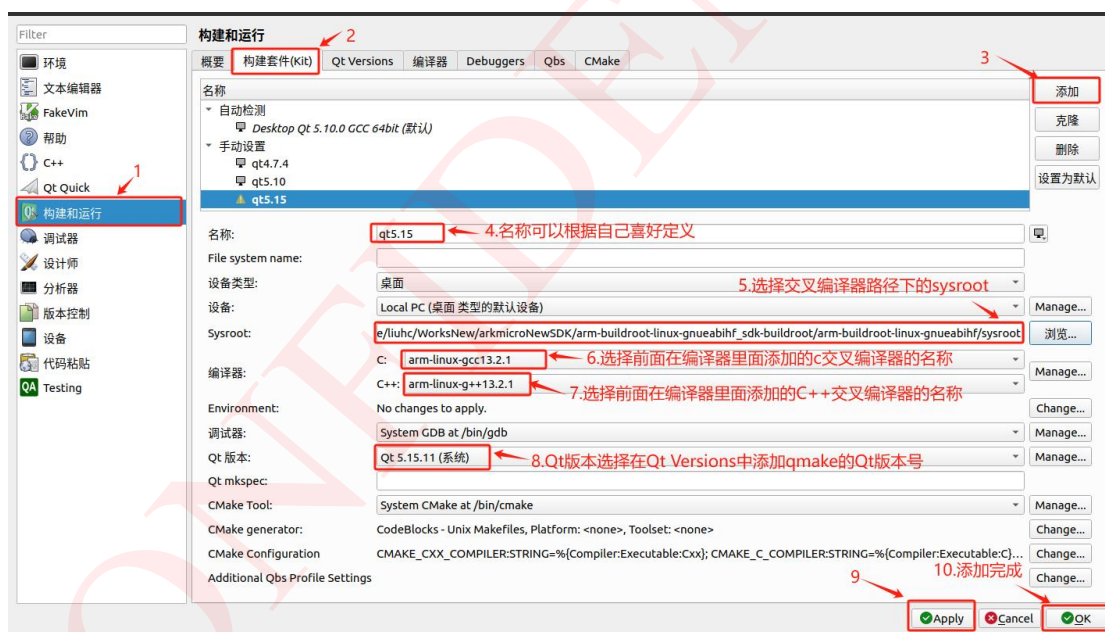
配置如下图：



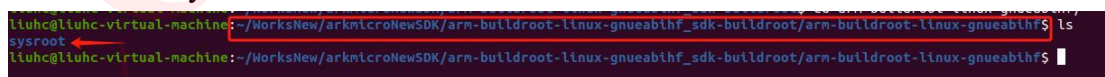
3.3.3 添加 qmake



3.3.4 配置构建套件



在此要注意 sysroot 路径是第一章中 1.2 步骤解压后生成的目录下。如下图：

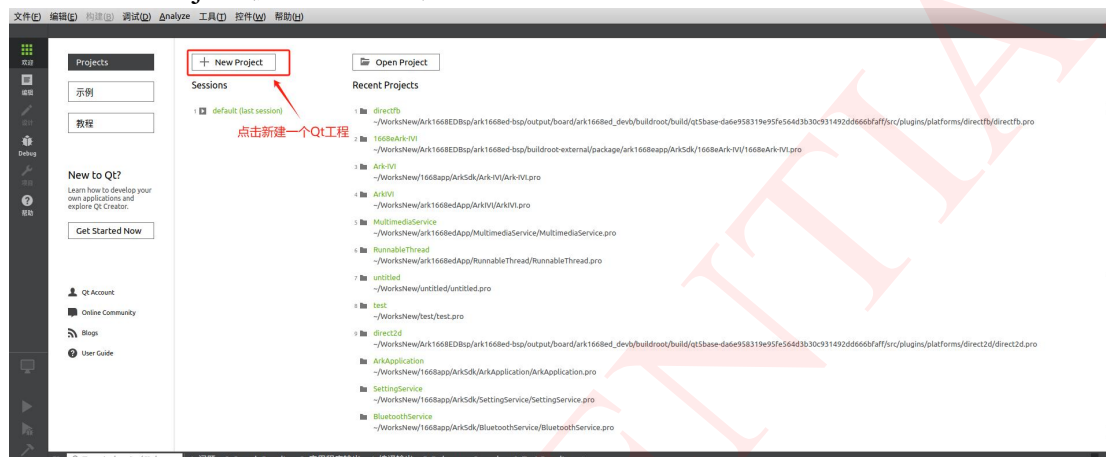


到此，Qt 交叉编译环境搭建完成！

第四章 新建 Qt 测试工程

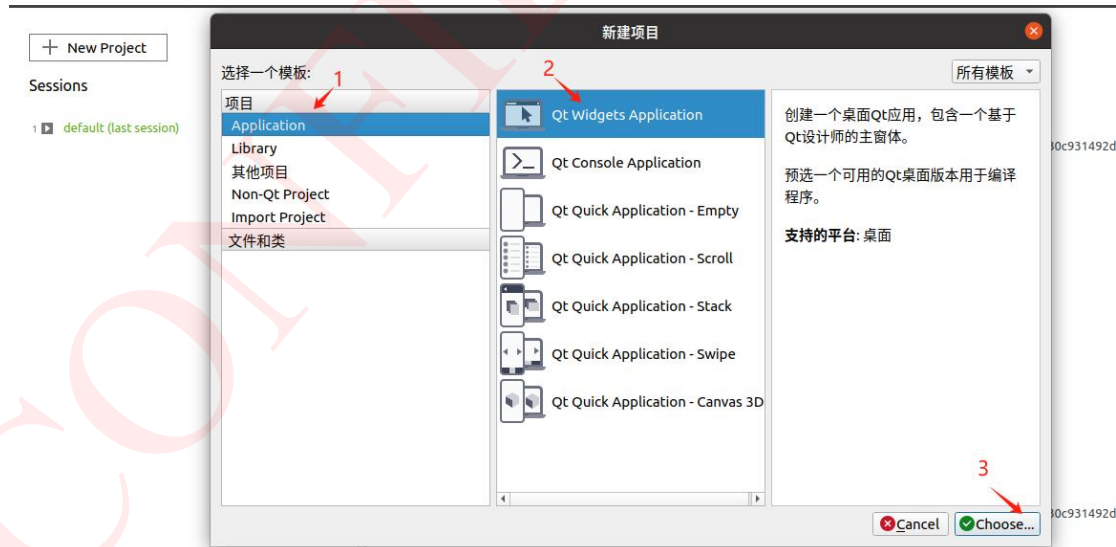
4.1 新建 Qt 工程

点击 New Project 按钮新建一个工程



4.2 配置工程

这里选择新建一个 QWidget 的应用



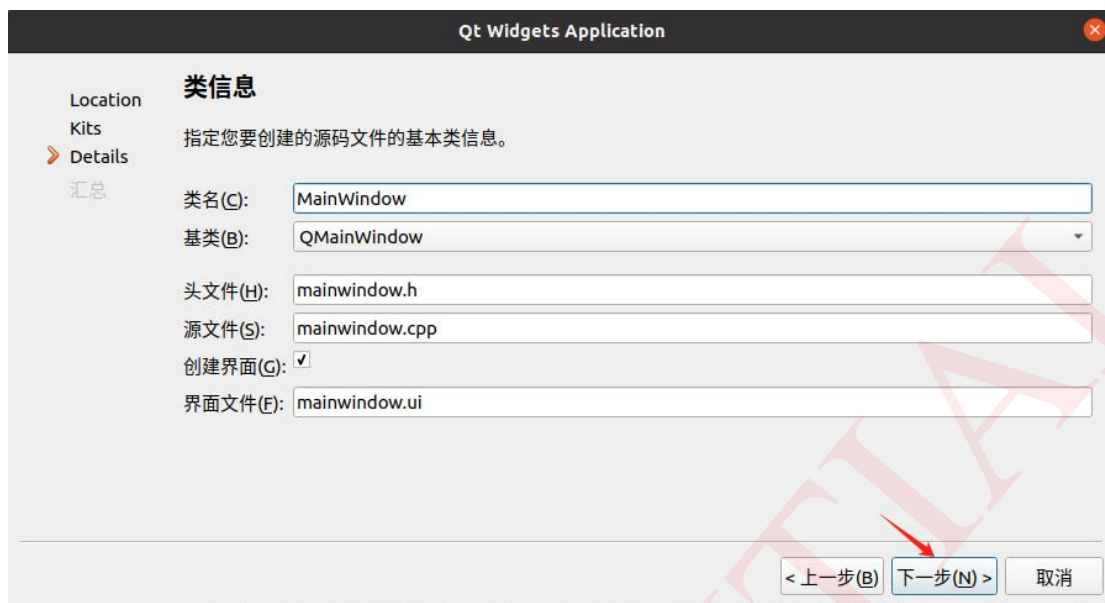
然后设置工程名字和工程路径。如下图：



选择构建套件为上面配置的交叉编译器套件



下面直接点击下一步

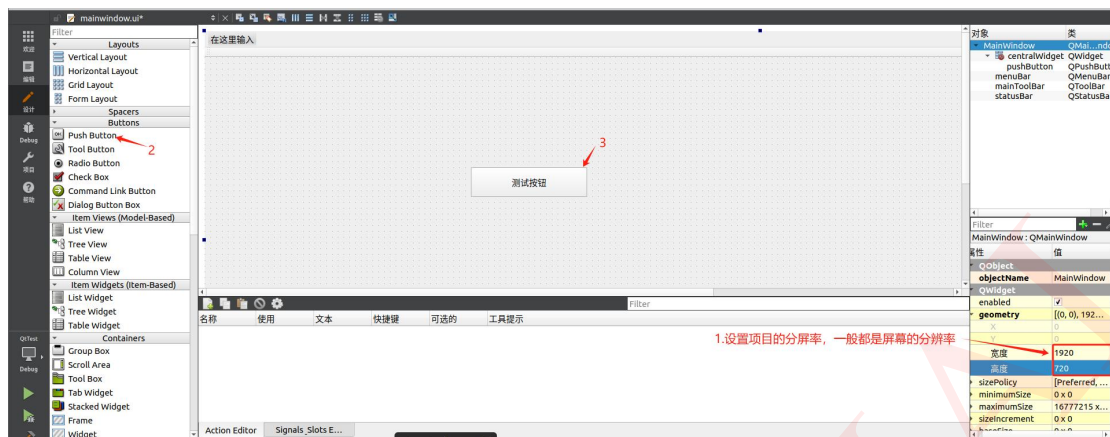


点击完成

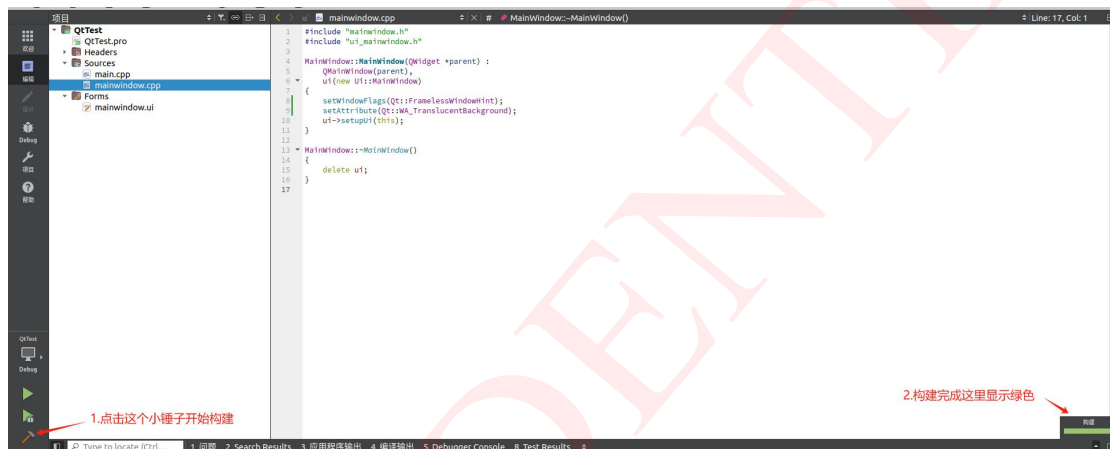


4.3 添加一个测试界面

可以随便设计一个界面，我这里如下



3.4 编译工程



若有报错提示没有安装 LGL，解决如下：

这是由于 Qt5.0 以上的版本默认将 OpenGL 加入了工程，但是在机器上没有安装 OpenGL，所以只需要在机器系统安装 OpenGL 即可

```
sudo apt-get install libgl1-mesa-dev
```

如果你不知道当前 Linux 系统中 libGL.so 的具体路径，可以使用 locate libGL 命令或 find /usr -name libGL* 命令查找，然后使用 ln -s 创建链接。请看下面的演示：

```
#查找 libGL 所在位置
[root@localhost ~]# locate libGL
/usr/lib64/libGL.so
/usr/lib64/libGL.so.1
/usr/lib64/libGL.so.1.2.0
/usr/share/doc/mesa-libGL-9.2.5
/usr/share/doc/mesa-libGL-9.2.5/COPYING

#创建链接
[root@localhost ~]# ln -s /usr/lib64/libGL.so.1 /usr/lib/libGL.so
```

Linux 系统中可能存在多个版本的 libGL.so，为任意一个版本创建链接即可。普通用户没有权限创建链接，所以我使用了 root 用户。

完成以上操作，再次启动 Qt，然后编译或者运行，就不会出现 “cannot find -lGL” 错误了。



第五章 在 Arm 平台运行 Qt 编译生成的应用

5.1 拷贝测试文件到 USB

切换到 QtTest 工程目录，拷贝编译生成的应用 QtTest 到 USB

```
liuhc@liuhc-virtual-machine:~/WorksNew/build-QtTest-qt5_15-Debug$ ls
main.o mainwindow.o Makefile moc_mainwindow.cpp moc_mainwindow.o moc_predefs.h QtTest ui_mainwindow.h
liuhc@liuhc-virtual-machine:~/WorksNew/build-QtTest-qt5_15-Debug$ pwd
/home/liuhc/WorksNew/build-QtTest-qt5_15-Debug
liuhc@liuhc-virtual-machine:~/WorksNew/build-QtTest-qt5_15-Debug$
```

拷贝这个可执行文件到U盘中

5.2 在开发板上运行测试文件

在开发板上运行 QtTest 应用，我们把 QtTest 拷贝到开发板的/usr/bin 目录下

```
#
# mount /dev/sda /mnt/
# cd /mnt/
# ls
1.gif
1080P
ARKSDLDR.bin
Launcher
MP3
QtTest
System volume Information
ark1668ed
ark1668ed_devb.dtb
bootanimation
crcdata.bin
filetest
galcore.ko
icon_120x120.png
# cp /mnt/QtTest /usr/bin/
# chmod 777 /usr/bin/QtTest
# vi /etc/init.d/S04qtlaunch
```

1
2
3
4

```
kbtest
keyBoardTest
libGAL.so
libQt5VirtualKeyboard.so.5.15.2
libdirectfb_gal.so
libqdirectfb.so
rootfs.ubi
u-boot.bin
u-boot.img
ubootspl.bin
update-magic
zImage
??????????
```

然后在/etc/init.d/S04qtlaunch 脚本中启动应用，重启机器，如果显示出你设计的界面的样式，则证明交叉编译环境搭建成功。

```
export QTDIR=/usr
export QT_QPA_FONTDIR=$QTDIR/lib/qt/fonts
#export QT_QPA_PLATFORM=linuxfb:fb=/dev/fb0
export QT_QPA_PLATFORM=directfb
export QT_QPA_PLATFORM_PLUGIN_PATH=$QTDIR/lib/qt/plugins
export QT_QPA_GENERIC_PLUGINS=evdevtouch:/dev/input/event0
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
#export ARKTVOUT=ENABLE
start() {
    export XDG_RUNTIME_DIR=/usr/lib/
    export RUNLEVEL=3
    eval dbus-launch --auto-syntax
    insmod /lib/modules/galcore.ko registerMemBase=0x40500000 contiguousSize=0x1000000 baseAddress=0x60000000 physSize=0x1c000000 powerManagement=0
    insmod /lib/modules/6.12.16/kernel/drivers/usb/dwc2/dwc2.ko
    carLink &
    Launcher &
    QtTest &
}
stop() {
    echo "stop"
}
restart() {
    stop
    sleep 1
    start
}
case "$1" in
    start|stop|restart)
        "$1";;
    reload)
        # Restart, since there is no true "reload" feature.
        restart;;
    *)
        echo "Usage: $0 {start|stop|restart|reload}"
        exit 1
esac
```

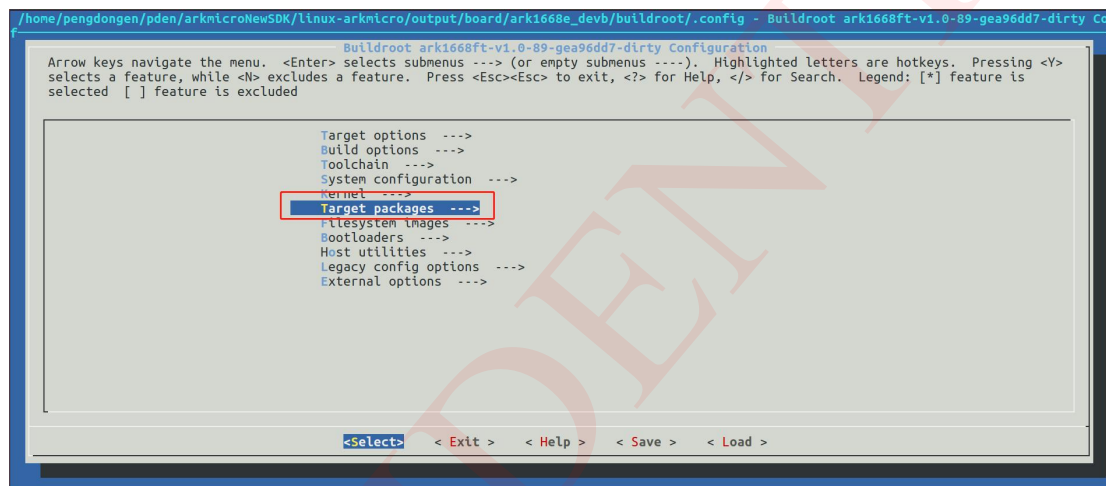
第六章 通过 Buildroot 裁剪 Qt 模块

6.1 重新配置 Buildroot 的 Qt 模块

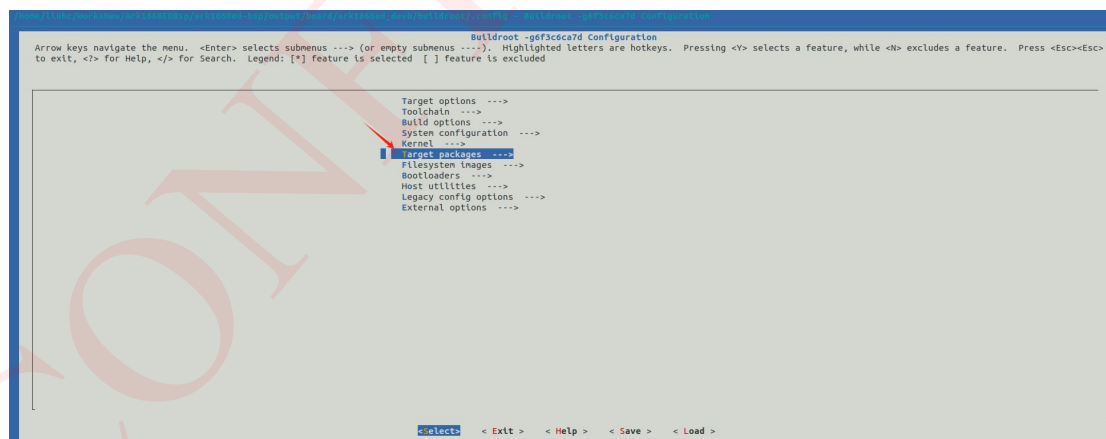
在 BSP 的 build 目录下打开终端执行下图中的指令配置 buildroot

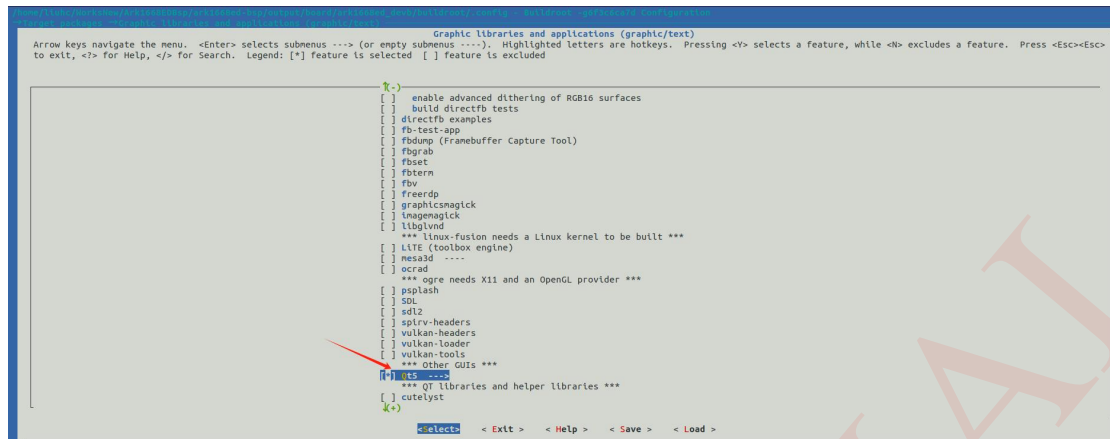
```
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp$ cd build
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$
liuhc@liuhc-virtual-machine:~/WorksNew/Ark1668EDBsp/ark1668ed-bsp/build$ ./build.sh ark1668ed_devb config buildroot
config buildroot...
# GEN      /home/liuhc/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/Makefile
#
# configuration written to /home/liuhc/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/.config
#
# GEN      /home/liuhc/WorksNew/Ark1668EDBsp/ark1668ed-bsp/output/board/ark1668ed_devb/buildroot/Makefile
```

在界面中选择 Target Packages 选项

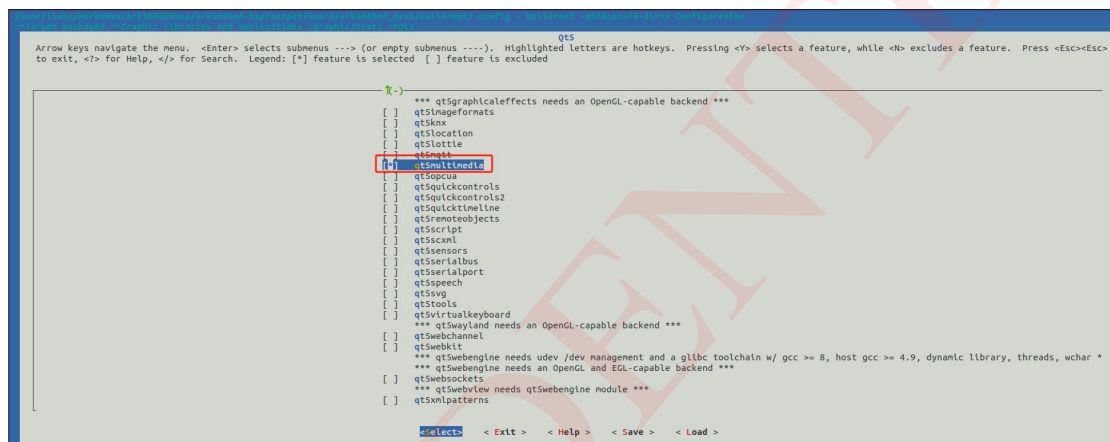


依次选择下图中的选项





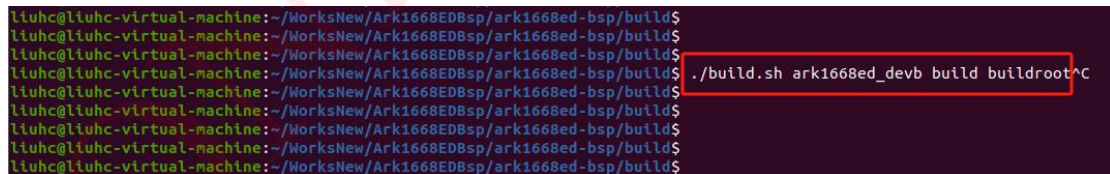
在 Qt5 选项中选择想要裁剪的 Qt 模块，这里选择添加多媒体模块，如下图：



添加完成后保存配置退出。

6.2 重新编译 buildroot

在 BSP 的 build 目录下执行如下图指令编译 buildroot



编译完成后拷贝升级文件到 U 盘后给平台重新升级。