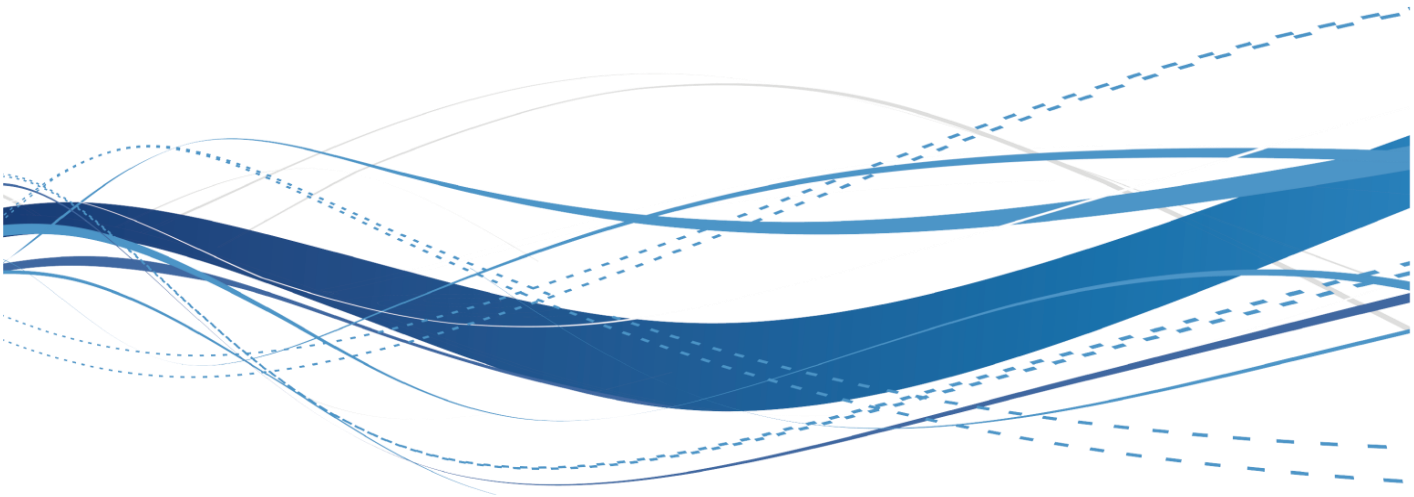
深圳市英蓓特科技公司

EM-TF-EVK-AM5728

Linux 软件开发指导

V1.0



版权声明

免责声明

版本记录

版本	描述	作者	日期
V1.0	初稿	David/Yuding	20180827

目录

第1章 环境搭建

1.1 搭建开发环境

搭建开发环境需要：

- 硬件：至少 20GB 磁盘空间，2GB 运行内存
- 软件：Ubuntu 64 bit 操作系统，14.04 LTS，16.04LTS 或更高的 LTS 版本 (Ubuntu Desktop 或 Ubuntu Server)

注：如需开发 QT，必须使用 Ubuntu Desktop 这类带 GUI 界面的 Ubuntu 版本。

也可以用虚拟机来运行 Ubuntu 64 bit 操作系统。

启动 Ubuntu 系统后，运行下列命令安装开发需要的软件。

```
sudo apt-get update
sudo apt-get install openssh-server
sudo apt-get install git
sudo apt-get install kpartx
sudo apt-get install lzop
sudo apt-get install lsb-core
sudo apt-get install libncurses5
```

1.2 配置编译环境

将发布文件夹的 gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz 拷贝到 Linux 环境下的$HOME 目录下，解压：

```
$tar -Jxvf gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz
```

设置环境变量：

```
$export
CROSS_COMPILE=$HOME/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
$export ARCH=arm
```

注意：每次编译 u-boot 和 kernel 前都要配置环境变量。为了方便起见，可以编辑一个脚本，然后 source 这个脚本：

```
$cd$HOME
$ cat set_am57_env.sh
#/bin/bash
export
CROSS_COMPILE=$HOME/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
export ARCH=arm
```

```
$ source ./set_am57_env.sh
```

注意：如未明确说明：

- 本文以$开头的命令行操作均指在 ubuntu PC 上；
- 文中出现的$HOME 没有强制要求，应以用户实际目录做相应的修改；
- 本文以#开头的命令行表示在开发板上执行。

1.3 其他工具和服务

开发过程中还可能会用到其他工具和服务，比如：

- ssh 登录 ubuntu/串口登录的 putty 软件
- 用于在 linux 和 windows 之间互传文件的 samba 服务
- NFS
- TFTP
- Samba

关于这些常规的开发工具的配置与使用，本文从略。

第2章 编译

2.1 U-Boot

2.1.1 获取 U-Boot 源码

复制 u-boot*.tar.gz 到$HOME 并解压：

```
$ cd $HOME
$ tar -xzvf u-boot*.tar.gz
```

2.1.2 编译 U-boot

```
$ cd $HOME/u-boot
$ make distclean
$make som_am572x_defconfig
$make
```

编译完成后在$HOME/u-boot 目录下生成 u-boot.img 和 MLO。

2.2 Kernel

2.2.1 获取内核源码

复制 linux 内核源代码包 linux*.tar.gz 到$HOME 并解压：

```
$ tar -zxvf linux*.tar.gz
```

2.2.2 编译镜像

```
$ cd $HOME/linux
$ make distclean
$ make embest_ti_am57xx_defconfig
$ make
```

编译完成生成 zImage 和 dtb：

- $HOME/linux/arch/arm/boot/zImage
- $HOME/linux/arch/arm/boot/dts/embest-SOM_AM572x_TM-mode0.dtb
- $HOME/linux/arch/arm/boot/dts/embest-SOM_AM572x_TM-mode0-LCD.dtb

其中 embest-SOM_AM572x_TM-mode0.dtb 将 HDMI 作为主显示屏，embest-SOM_AM572x_TM-mode0-LCD.dtb 将 LCD 作为主显示屏。

2.3 外部驱动

由于 TI 的部分外设模块的驱动是单独发布的，因此这部分驱动程序需要额外编译。这些外设模块包括 2/3D 图像加速模块，硬件加/解密模块等。复制 extra.tar.gz 到$HOME 并解压

```
$ cd $HOME/
$ tar -xzf extra.tar.gz
```

2.3.1 配置环境变量

编辑 Rules.make 文件，修改以下几个变量为相应的值：

- DESTDIR 根文件系统所在路径，如已经烧写镜像的 SD 卡在 ubuntu 下挂载路径，也可以是任意其他空白目录。
- CROSS_COMPILE 为交叉编译工具链所在路径
- LINUXKERNEL_INSTALL_DIR 为 linux 内核源代码路径，在编译外部驱动之前，先确保 linux 内核源代码被正确地配置和编译，参见 2.2Kernel 的编译。

```
$ cd extra
$ ls
extra-drivers    MakefileRules.make
$cat Rules.make
#platform
#platform
PLATFORM=am57xx-evm

#root of the target file system for installing applications
DESTDIR=$(HOME)/extra/fakeroot

#Cross compiler prefix
export
CROSS_COMPILE=$(HOME)/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

#The directory that points to the SDK kernel source tree
LINUXKERNEL_INSTALL_DIR=$(HOME)/linux

CFLAGS= -march=armv7-a -marm -mfpu=neon    -mfloat-abi=hard

#Strip modules when installing to conserve disk space
INSTALL_MOD_STRIP=1

export TOOLCHAIN_PREFIX=$(CROSS_COMPILE)
```

2.3.2 编译外部驱动

编译之前，先创建 Rules.make 中 DESTDIR 变量指定的空白目录。此处我们创建 fakeroot 目录

```
$ cd extra
$ mkdir fakeroot
$ ls
extra-drivers  fakeroot  Makefile  Rules.make
$ make clean
$ make
```

2.3.3 安装外部驱动

```
$ make install
```

编译生成的 ko 文件以及相关的文件会被安装在 fakeroot 下，如果只需要更新 ko 文件，则只需要复制 lib/modules/4.9.28/extra 下的 ko 文件到目标板文件系统的相应目录下即可。

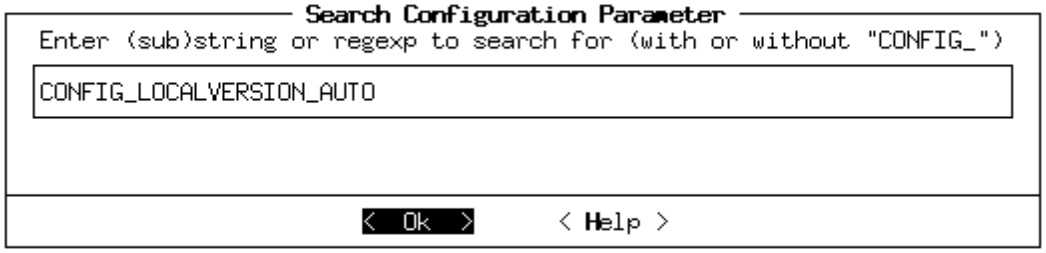
```
$ tree fakeroot/
fakeroot/
├── lib
│   ├── firmware
│   │   └── jailhouse.bin
│   └── modules
│       └── 4.9.28
│           ├── extra
│           │   ├── bc_example.ko
│           │   ├── cmemk.ko
│           │   ├── cryptodev.ko
│           │   ├── debugss_kmodule.ko
│           │   ├── driver
│           │   │   └── jailhouse.ko
│           │   ├── galcore.ko
│           │   ├── gdbserverproxy.ko
│           │   ├── pvrsrvkm.ko
│           │   └── uio_module_drv.ko
│           ├── modules.alias
│           ├── modules.alias.bin
│           ├── modules.builtin.bin
│           ├── modules.dep
│           ├── modules.dep.bin
│           ├── modules.devname
│           ├── modules.softdep
```

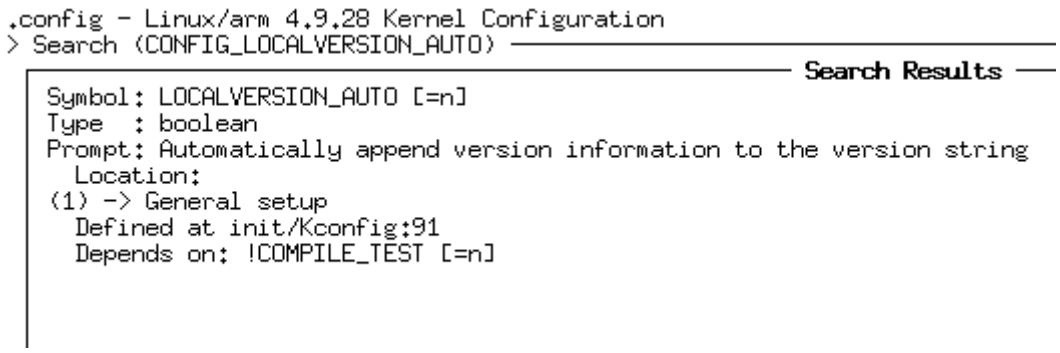
```
|               ├──modules.symbols
|               └──modules.symbols.bin
└──usr
├──libexec
|   └── jailhouse
|       ├── jailhouse-cell-linux
|       ├── jailhouse-cell-stats
|       ├── jailhouse-config-create
|       ├── jailhouse-hardware-check
|       └──linux-loader.bin
├──sbin
|   └── jailhouse
└── share
├── bash-completion
|   └── completions
|       └── jailhouse
└── jailhouse
├── jailhouse-config-collect.tmpl
└── root-cell-config.c.tmpl

14 directories, 28 files
```

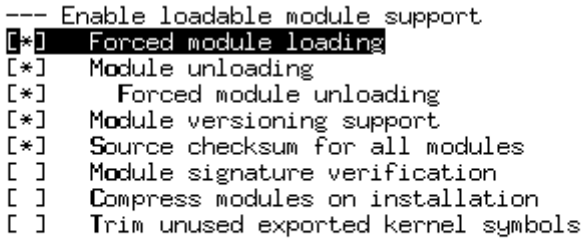
如果为了开发方便，可关闭内核中的 git 版本控制选项。以后编译出来的内核版本都是 linux-4.9.28，因此不用反复的编译 extra 相关的驱动模块。

在 kernel 根目录输入 make menuconfig，打开配置界面，在 menuconfig 界面键入/，搜索 CONFIG_LOCALVERSION_AUTO 选项，根据提示找到相关选项，将其设置为[=n]:





在开发阶段，也可以使能 CONFIG_MODULE_FORCE_LOAD。



第3章 制作镜像

3.1 制作镜像文件

发布文件夹中有做好的镜像文件（如 EM-TF-EVK-AM5728-TI-ShipmentImage-SDcard-V1.0.3r04.img）可供用户用 dd 命令（linux 环境）和 Win32DiskImager.exe（Windows 环境）烧录到 SD 卡或者 EMMC 中，具体请参考用户手册。这一节我们将介绍如何将上一节中编译得到的产物做成一个可供烧录使用的镜像文件。

我们制作的这个镜像文件大小不能超过准备烧录的 SD 卡/EMMC 容量的大小。假设 SD 卡为 4GB，我们可以创建一个 3800MB 的空白磁盘文件，文件名为 example.img，在 ubuntu 下：

```
$ cd $HOME
$ sudo dd if=/dev/zero of=./example.img bs=1M count=3800
3800+0 records in
3800+0 records out
3984588800 bytes (4.0 GB, 3.7 GiB) copied, 9.85035 s, 405 MB/s
```

3.1.1 分区格式化

将 example.img 分为两个区。第一个区为 FAT32，大小 64MB，用于存放 firmware；第二个分区为 ext4 格式用于存放根文件系统。

```
$ cd $HOME
$ sudo fdisk example.img

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x928aa0d6.

Command (m for help): o
Created a new DOS disklabel with disk identifier 0x12076151.

Command (m for help): n
Partition type
p   primary (0 primary, 0 extended, 4 free)
e   extended (container for logical partitions)
Select (default p): p
```

```
Partition number (1-4, default 1): 1
First sector (2048-7782399, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-7782399, default 7782399): +64M

Created a new partition 1 of type 'Linux' and of size 64 MiB.

Command (m for help): t
Selected partition 1
Partition type (type L to list all types): c
Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.

Command (m for help): a
Selected partition 1
The bootable flag on partition 1 is enabled now.

Command (m for help): n
Partition type
p   primary (1 primary, 0 extended, 3 free)
e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2):
First sector (133120-7782399, default 133120):
Last sector, +sectors or +size{K,M,G,T,P} (133120-7782399, default 7782399):

Created a new partition 2 of type 'Linux' and of size 3.7 GiB.

Command (m for help): p
Disk example.img: 3.7 GiB, 3984588800 bytes, 7782400 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x12076151

Device          Boot  Start     End Sectors  Size Id Type
example.img1 *        2048  133119  131072   64M  c W95 FAT32 (LBA)
example.img2        133120 7782399 7649280  3.7G 83 Linux
```

```
Command (m for help): w
The partition table has been altered.
Syncing disks.
```

格式化 FAT32 分区并设置卷标为 boot，格式化 ext4 分区并设置卷标为 rootfs

```
$ sudo losetup /dev/loop0 example.img
$ sudo kpartx -av /dev/loop0
add map loop0p1 (253:0): 0 131072 linear 7:0 2048
add map loop0p2 (253:1): 0 7649280 linear 7:0 133120
$ ls /dev/mapper/loop0p*
/dev/mapper/loop0p1    /dev/mapper/loop0p2
$ sudo mkfs.vfat -F 32 -n "boot" /dev/mapper/loop0p1
mkfs.fat 3.0.28 (2015-05-16)
mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows
unable to get drive geometry, using default 255/63
$ sudo mkfs.ext4 -L "rootfs" /dev/mapper/loop0p2
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 956160 4k blocks and 239040 inodes
Filesystem UUID: 0820d179-521d-4f91-816f-df13309eee87
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

3.1.2 复制 firmware

创建两个临时目录，分别用于挂载

```
$ mkdir boot rootfs
$ sudo mount /dev/mapper/loop0p1 boot/
$ sudo mount /dev/mapper/loop0p2 rootfs/
```

通过 lsblk 命令可以查看挂载是否成功

```
$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0         7:0    0   3.7G  0 loop
├─loop0p2 253:1    0   3.7G  0 part /home/david/rootfs
└─loop0p1 253:0    0    64M  0 part /home/david/boot
```

```
$ sudo cp $HOME/u-boot/u-boot.img ./boot
$ sudo cp $HOME/u-boot/MLO./boot
$ sudo cp $HOME/linux/arch/arm/boot/zImage./boot
$ sudo cp $HOME/linux/arch/arm/boot/dts/embest-SOM_AM572x_TM-mode0.dtb ./boot
$ sudo cp $HOME/linux/arch/arm/boot/dts/embest-SOM_AM572x_TM-mode0-LCD.dtb ./boot
```

在./boot 中创建 uEnv.txt 文件，指定 dtb 文件，例如以 HDMI 作为主显示屏

```
$ sudo touch ./boot/uEnv.txt
$ sudo bash -c "echo fdtfile=embest-SOM_AM572x_TM-mode0.dtb > ./boot/uEnv.txt"
```

3.1.3 复制根文件系统

解压 tisdk-rootfs-image-am57xx-evm.tar.xz 到$HOME/rootfs-arago，复制 tisdk-rootfs-image-am57xx-evm.tar.xz 中全部内容到 rootfs 目录，注意此处需要使用 cp -ap 选项，以确保文件的属性保持不变。

```
$ mkdir rootfs-arago
$ tar -Jxvf tisdk-rootfs-image-am57xx-evm.tar.xz -C rootfs-arago/
$ sudo cp -ap rootfs-arago/* rootfs
```

3.1.4 安装内核模块

安装内核模块需要 root 权限，因此也需要给 root 用户设置环境变量

```
$ cd $HOME/linux
$ sudo make modules_install INSTALL_MOD_PATH=$HOME/rootfs ARCH=arm
CROSS_COMPILE=$HOME/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

3.1.5 安装外部驱动模块

参见 2.3.3 安装外部驱动模块，将 DESTDIR 变量设置为$HOME/rootfs，然后安装即可。

如果用户没有升级内核版本的情况下，我们提供的 rootfs-arago.tar.gz 根文件系统中已经包含了内核模块以及外部驱动模块，因此不需要重复安装。

拷贝完成后卸载两个分区，并同步文件系统

```
$ sudo umount boot rootfs
$ sudo kpartx -d /dev/loop0
$ sudo losetup -d /dev/loop0
$ sync
```

3.2 烧录和读取镜像

3.2.1 烧录

参见用户手册。

3.2.2 读取

开发过程中常需要将 SD 卡中镜像读取出来备份，可以使用如下命令得到 SD 卡的镜像文件

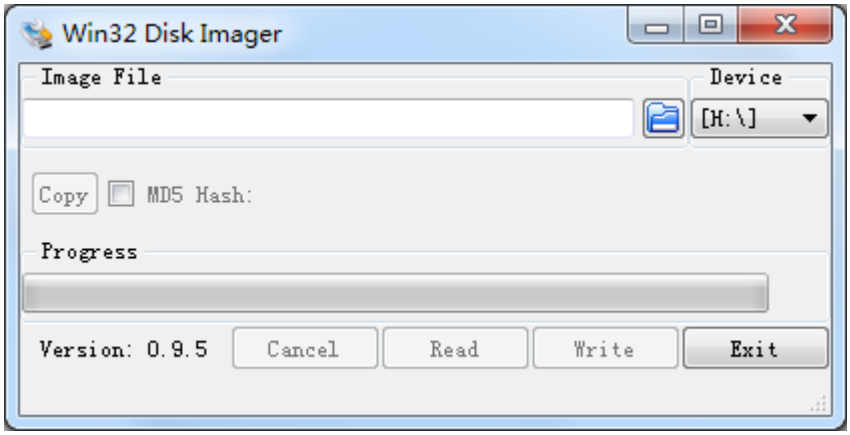
- 在 linux 环境下

将 SD 卡接入读卡器连接到电脑

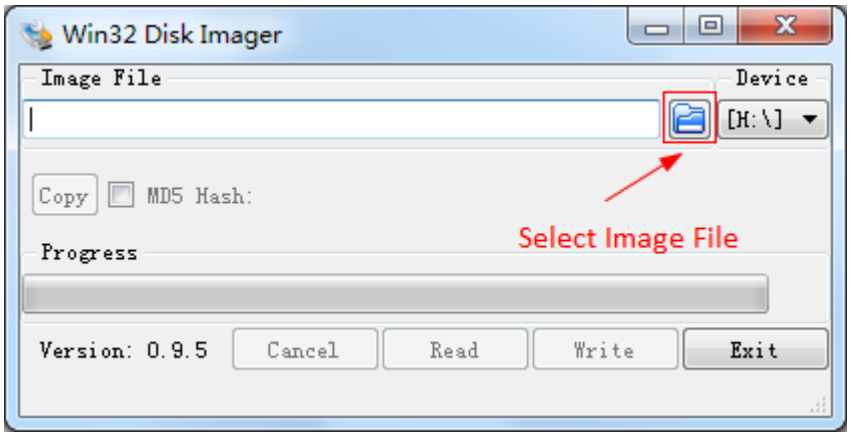
```
~/ti/linux$ lsblk
NAME     MAJ:MIN RM     SIZE RO TYPE MOUNTPOINT
sdb        8:16     1     7.2G    0 disk
 ├─sdb2    8:18     1     3.8G    0 part
 └─sdb1    8:17     1      64M    0 part
$ dd if=/dev/sdb of=./sdcard.img
```

- 在 windows 环境下

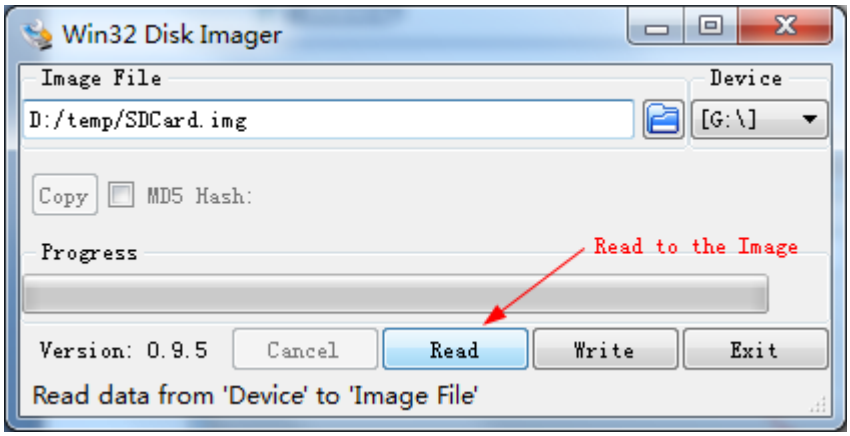
将 SD 卡连接到电脑，运行 Win32 Disk Imager



选择镜像文件的存放地址，如：D:/temp/SDCard.img



点击 Read 将 SD 卡的内容读取到镜像文件中：



执行成功后，你会获得一个完整的 SD 镜像。

第4章 TI SDK 开发

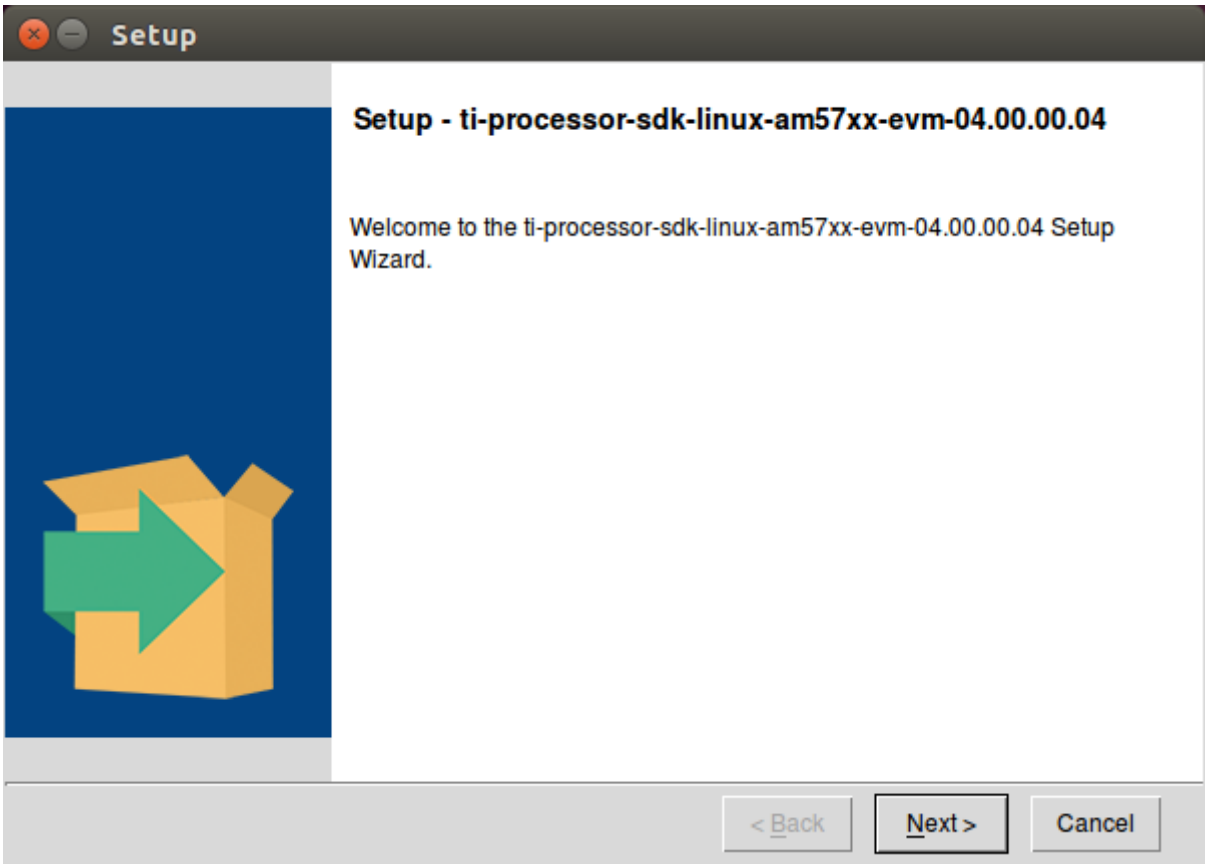
4.1 SDK 的安装和配置

TI SDK 安装包可以在 ssh 命令行安装，也可以在 ubuntu 桌面环境安装，一下我们以桌面环境安装为例：

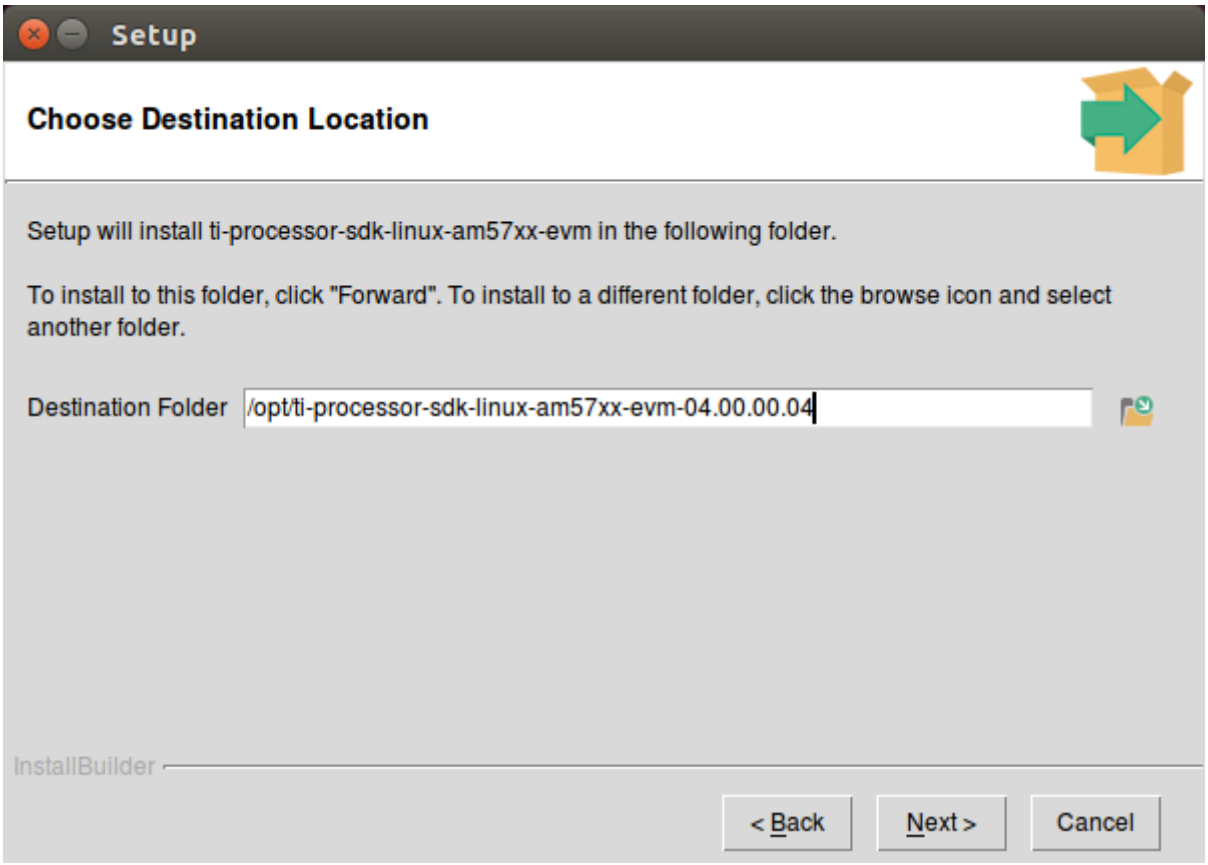
在 ubuntu 桌面按组合键 Ctrl+Alt+T 打开控制台

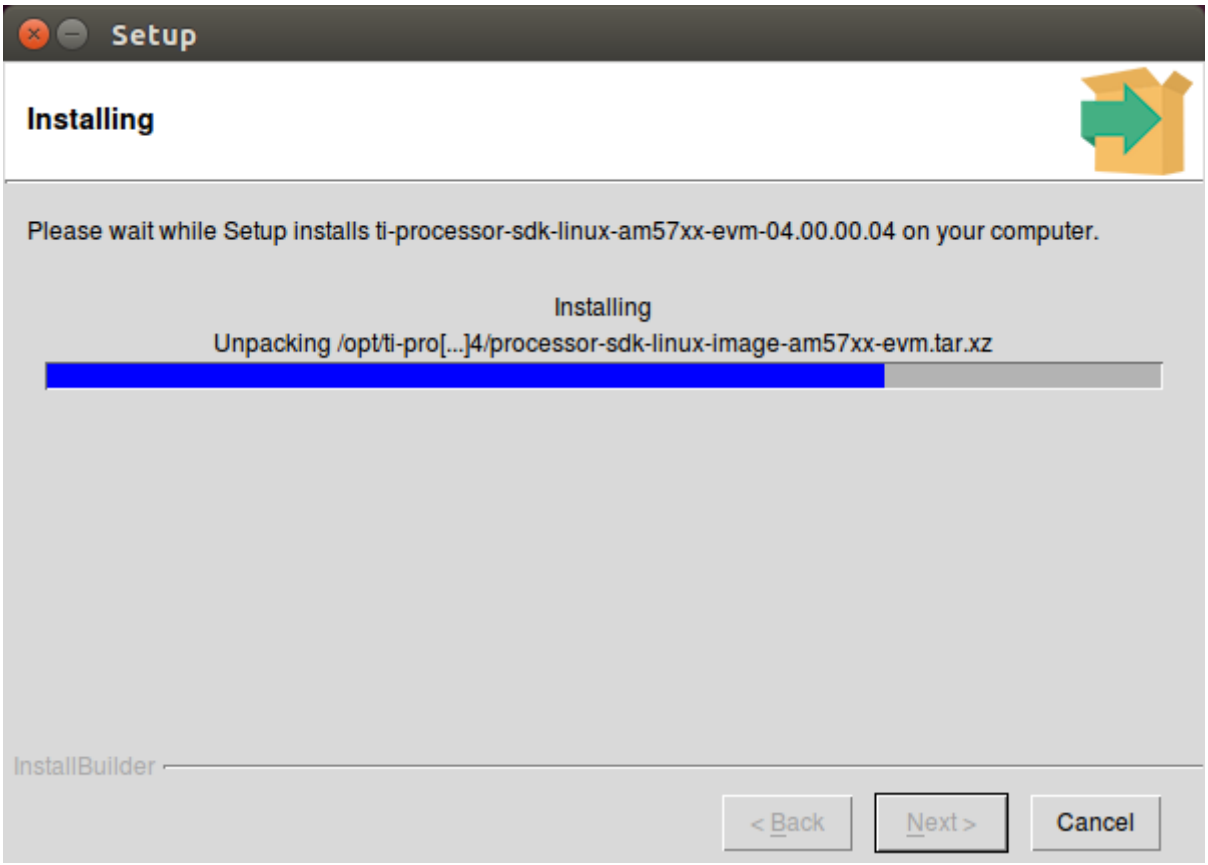
```
$ cd $HOME
$ sudo chmod +x ti-processor-sdk-linux-am57xx-evm-04.00.00.04-Linux-x86-Install.bin
$ sudo ./ti-processor-sdk-linux-am57xx-evm-04.00.00.04-Linux-x86-Install.bin
```

弹出窗口

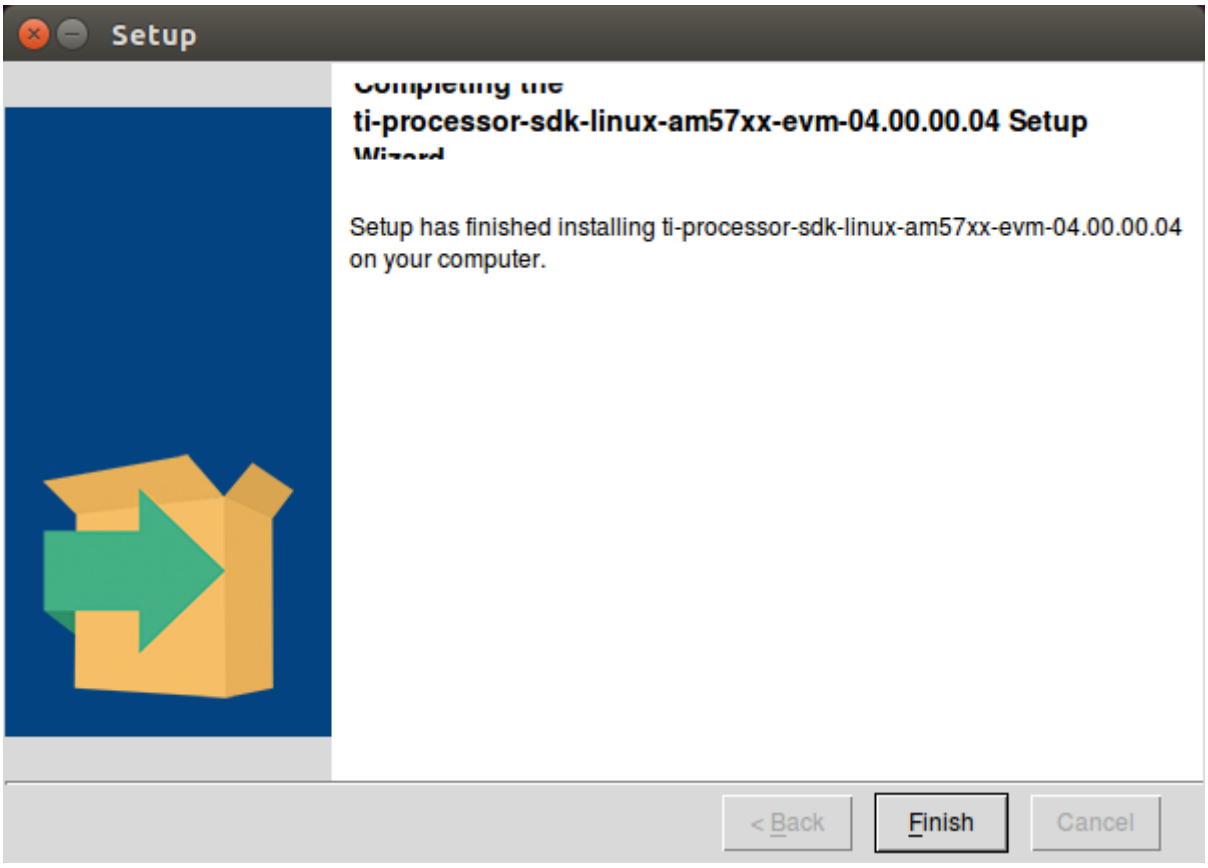


根据指引信息，点击下一步，出现选择安装路径时，可以使用默认路径安装，也可以自定义安装路径





大约几分钟后，安装完成，点击“Finish”



TI 的 SDK 目录组织如下：

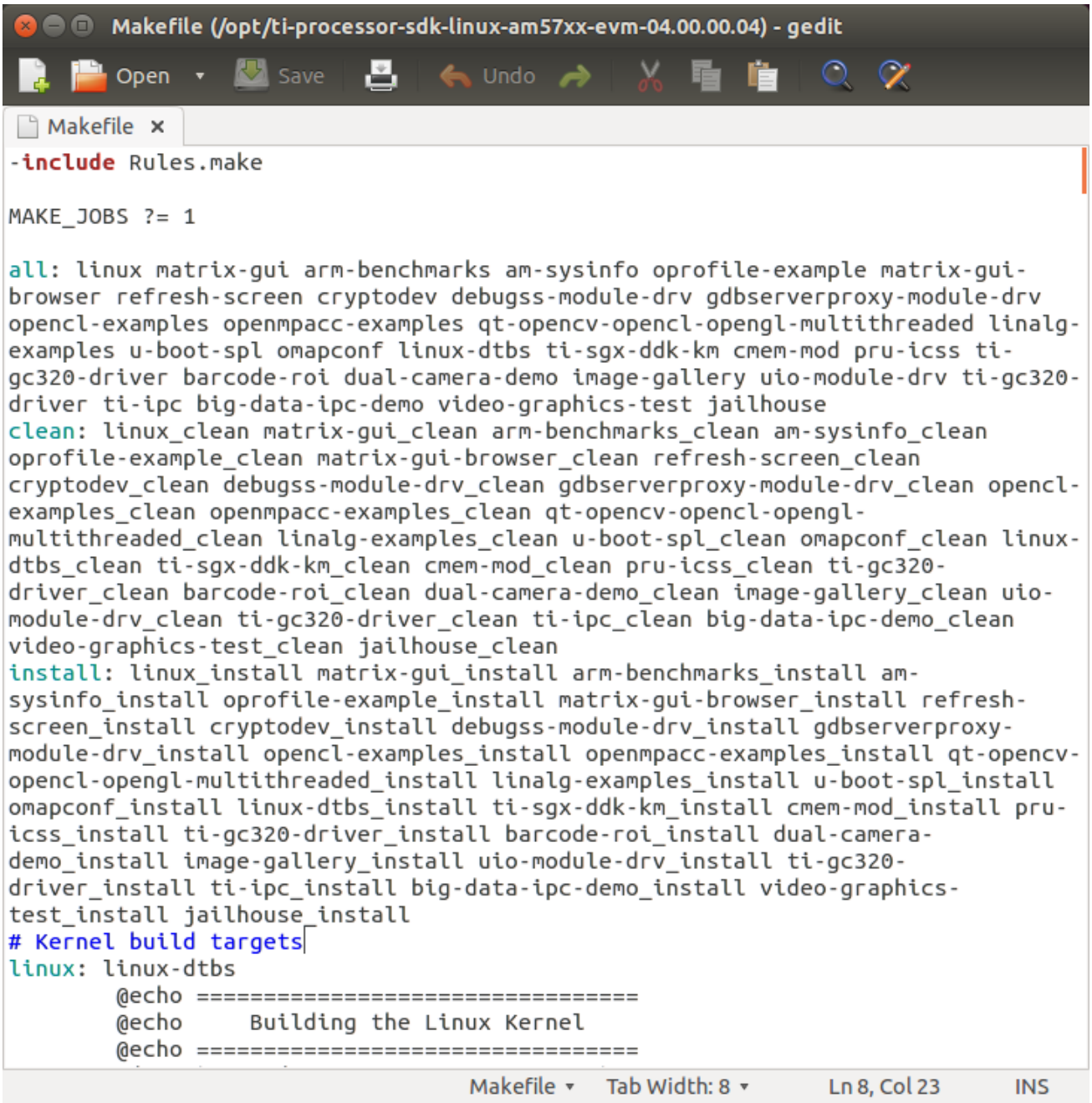
```
$ cd /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/
$ ls -l
total 542220
drwxr-xr-x  2smbusersmbuser         4096 Aug 10 16:18 bin
drwxr-xr-x  6smbusersmbuser         4096 Jun 29  2017 board-support
drwxr-xr-x  3smbusersmbuser         4096 Jun 29  2017 docs
drwxr-xr-x 19 smbusersmbuser        4096 Jun 29  2017 example-applications
drwxr-xr-x  2smbusersmbuser         4096 Jun 29  2017 filesystem
drwxr-xr-x  3 root       root       4096 Aug 10 16:18 linux-devkit
-rwxr-xr-x  1 smbusersmbuser 555147047 Jun 29  2017 linux-devkit.sh
-rwxr-xr-x  1 smbusersmbuser        44597 Jun 29  2017 Makefile
-rwxr-xr-x  1 smbusersmbuser         1324 Aug 10 16:18 Rules.make
-rwxr-xr-x  1 smbusersmbuser         4188 Jun 29  2017 setup.sh
```

其各个目录/文件用途如下：

bin	一些工具脚本，用于制作 SD 卡，设置 TFTP 等
board-support	1， u-boot 源代码； 2， Kernel 源代码； 3， 外部驱动源代码； 4， 针对 TI 的评估板而预编译好的 firmware
docs	SDK 软件列表以及 license
example-applications	Demo 程序源代码
filesystem	基于 Arago 的预构建好的根文件系统
linux-devkit	编译整个 SDK 需要的根文件系统，其中包含了交叉编译工具链
linux-devkit.sh	linux-devkit 文件夹的压缩包，移动已经安装好的 SDK 路径后，需要这个文件
Makefile	顶层 Makefile 文件，可以编译整个 SDK，包括： 1， Linux Kernel 2， U-boot 3， 外部驱动 4， Demo 程序
Rules.make	编译 SDK 需要用到的一些环境变量，如交叉编辑工具链，目标根文件系统路径
setup.sh	设置开发环境，实际是调用 bin 目录下的一些工具脚本

4.2 顶层 Makefile 使用

SDK 中的顶层 Makefile 中包含了很多目标，通过这个 Makefile 可以全部编译 all，clean，install，也可以单独指定某个目标进行编译，清除，和安装。



4.2.1 编译整个 SDK

顶层 Makefile 默认目标 all 可以编译整个 SDK，因此只需要一条命令即可：

```
$ sudo make
```

编译过程从第一个目标 linux 开始解析，而 linux 依赖的第一个目标是 linux-dtbs，因此最先执行的编译目标是 linux-debs。需要一定的时间，编译完成后如需安装编译产物到目标文件系统，则需要修改 Rule.make 中的 DESTDIR 为实际根文件系统所在目录。此处设置 DESTDIR=/opt/fakeroot，首先要先创建这个目录，然后 make install。

```
$ cd /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04
$ sudo make install
$ cd ../fakeroot
$ ls -l
total 12
drwxr-xr-x 2 root root 4096 Aug 10 17:39 boot
drwxr-xr-x 4 root root 4096 Aug 10 17:39 lib
drwxr-xr-x 4 root root 4096 Aug 10 17:39 usr
```

4.2.2 单独编译安装某个目标

以编译 dual-camera 这个 demo 程序为例：

```
$ cd /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04
$ sudo make dual-camera-demo
Makefile:715: warning: overriding commands for target `ti-gc320-driver'
Makefile:595: warning: ignoring old commands for target `ti-gc320-driver'
Makefile:723: warning: overriding commands for target `ti-gc320-driver_clean'
Makefile:603: warning: ignoring old commands for target `ti-gc320-driver_clean'
Makefile:731: warning: overriding commands for target `ti-gc320-driver_install'
Makefile:611: warning: ignoring old commands for target `ti-gc320-driver_install'
================================
Building Dual Camera Demo
================================
make[1]: Entering directory
`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'
echo "manisha" am57xx-evm
manisha am57xx-evm
make[2]: Entering directory
`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'
make[2]: Nothing to be done for `first'.
make[2]: Leaving directory
```

```
`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'
make[1]: Leaving directory
`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'
david@ubuntu:/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04$
```

安装

```
$ sudo make dual-camera-demo_install
Makefile:715: warning: overriding commands for target `ti-gc320-driver'
Makefile:595: warning: ignoring old commands for target `ti-gc320-driver'
Makefile:723: warning: overriding commands for target `ti-gc320-driver_clean'
Makefile:603: warning: ignoring old commands for target `ti-gc320-driver_clean'
Makefile:731: warning: overriding commands for target `ti-gc320-driver_install'
Makefile:611: warning: ignoring old commands for target `ti-gc320-driver_install'
================================================
Installing Dual Camera Demo - Release version
================================================
make[1]: Entering directory
`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'
echo "manisha" am57xx-evm
manisha am57xx-evm
make[2]: Entering directory
`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'
make[2]: Nothing to be done for `first'.
make[2]: Leaving directory
`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'
dual_camera release version installed.
make[1]: Leaving directory
`/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/example-applications/dual-camera-demo-1.0'
```

第5章 应用开发

5.1 普通 C 程序的交叉编译和执行

5.1.1 普通 C 程序的交叉编译和执行

解压 application.tar.gz 到$HOME，以编译 4G 通信测试程序为例：

5.1.2 交叉编译

```
$ cd $HOME/application/4g_test
$ $HOME/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc 4G_test.c -o 4G_test
$ ls
4G_test   4G_test.c   readme.md
$ file 4G_test
4G_test: ELF 64-bit LSB   executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=69a5b54a2de0c56b075f871fff6710797250a72c, not stripped
```

5.1.3 直接编译

我们发布的根文件系统中已经安装了编译工具链，因此可以复制 c 源代码文件到直接在板上使用 gcc 编译应用程序

```
# gcc 4G_test.c -o 4G_test
# ls
4G_test   4G_test.
```

5.1.4 传递到开发板运行

- 将开发板与电脑接入统一局域网，使用 scp 命令传递文件

```
# scp $HOME/application_test_programs/4g_test/4g_test ./
```

- 使用 U 盘等存储介质拷贝

略

5.2 QT 开发

5.2.1 安装 Qt Creator

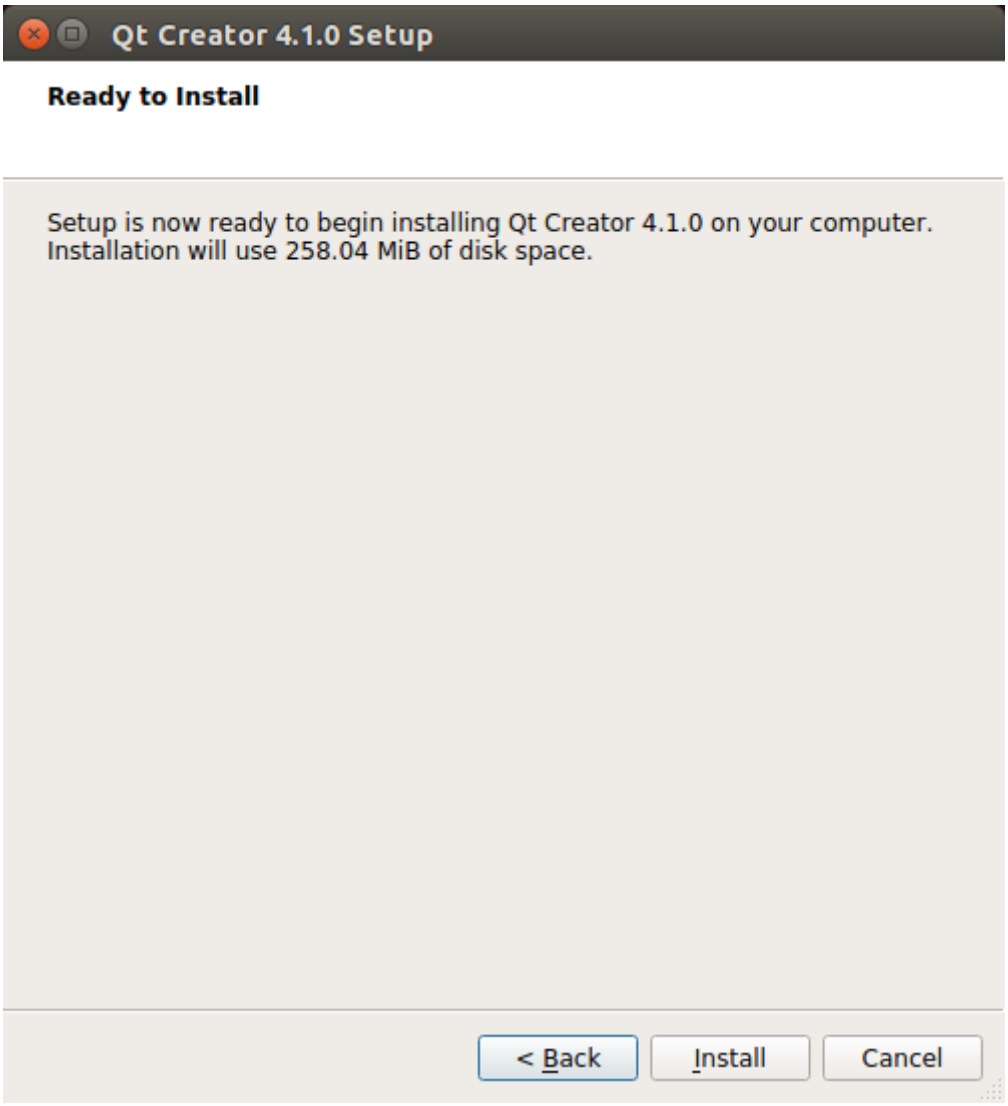
Qt Creator 是一个图形化的设计器，因此这一部分的操作均在 ubuntu 桌面环境下进行。拷贝 qt-creator-opensource-linux-x86_64-4.1.0.run 到$HOME，增加执行权限。

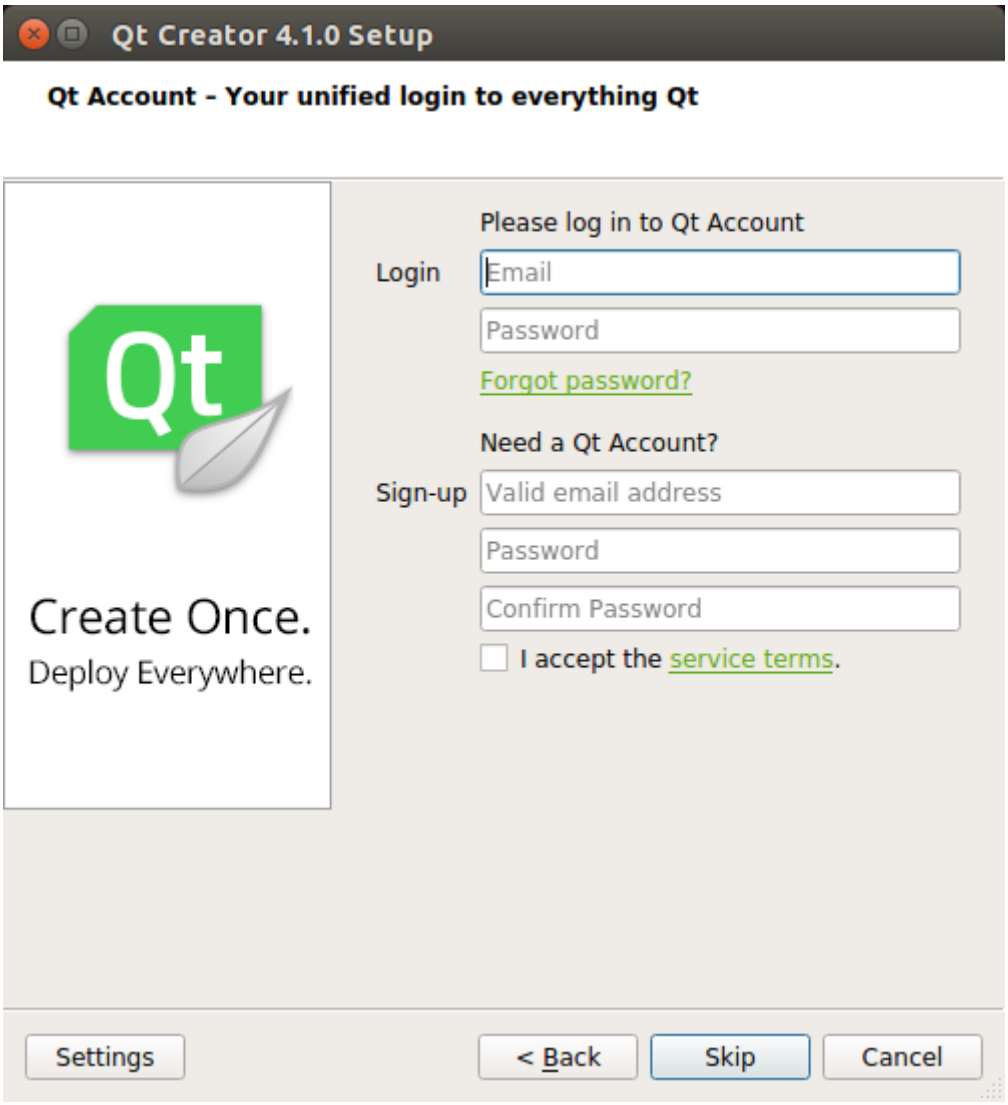
```
$ sudo chmod +x qt-creator-opensource-linux-x86_64-4.1.0.run
```

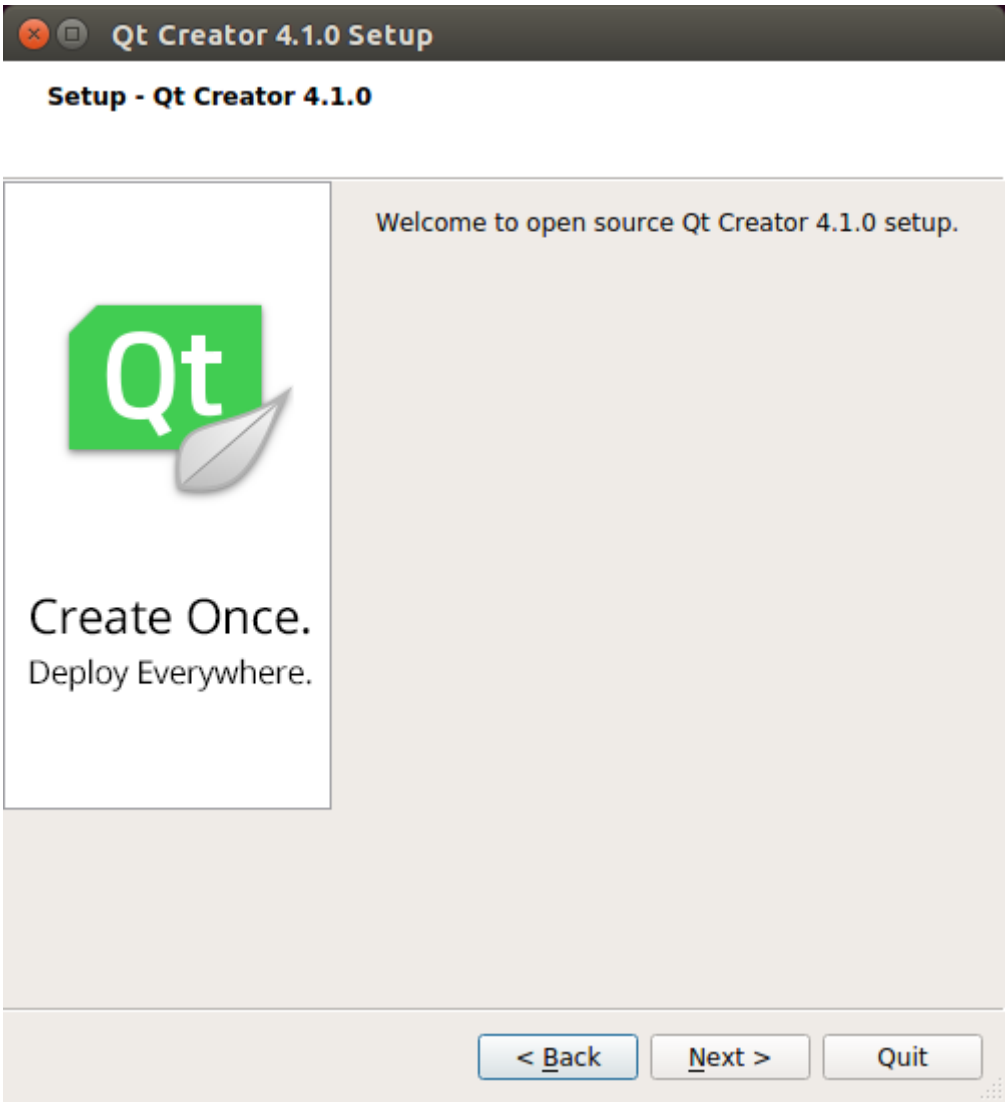
执行安装程序

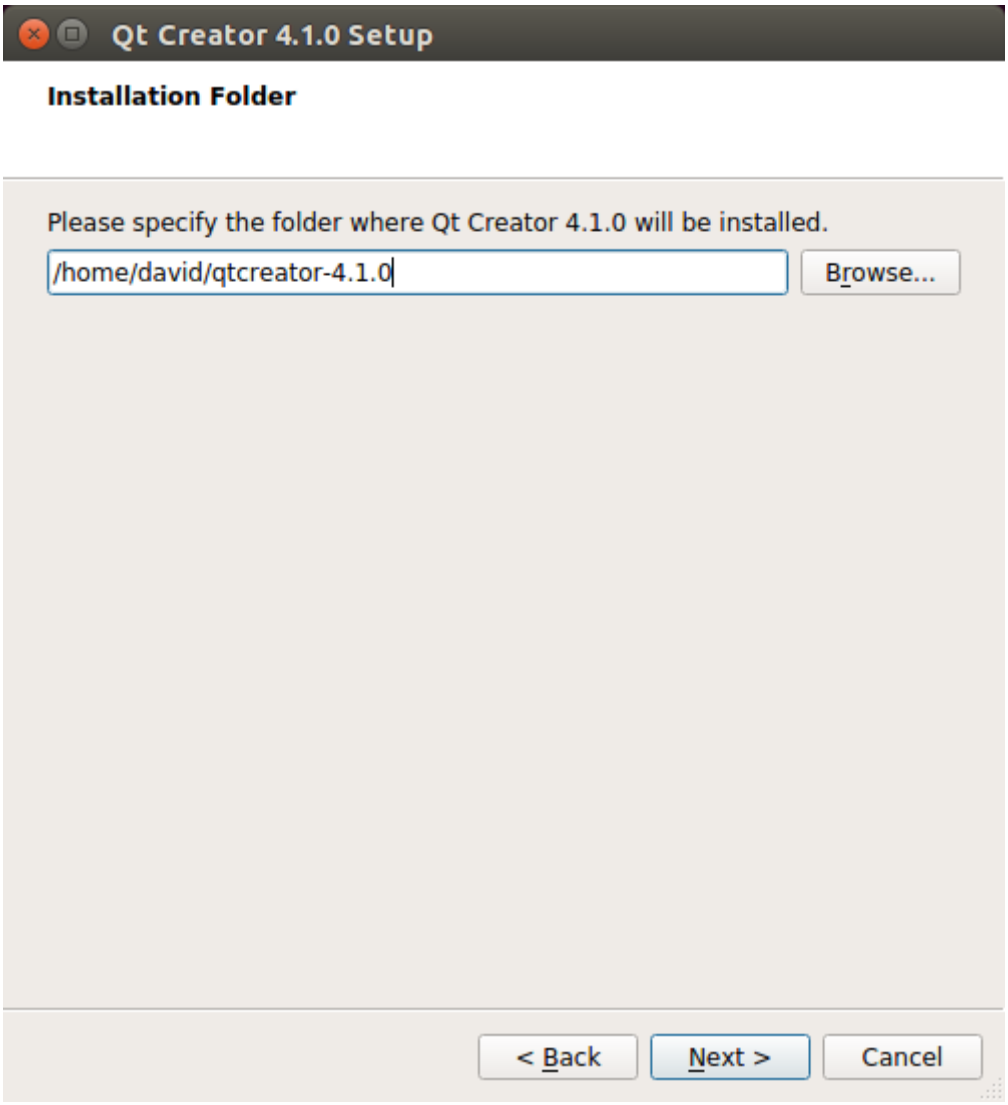
```
$ ./qt-creator-opensource-linux-x86_64-4.1.0.run
```

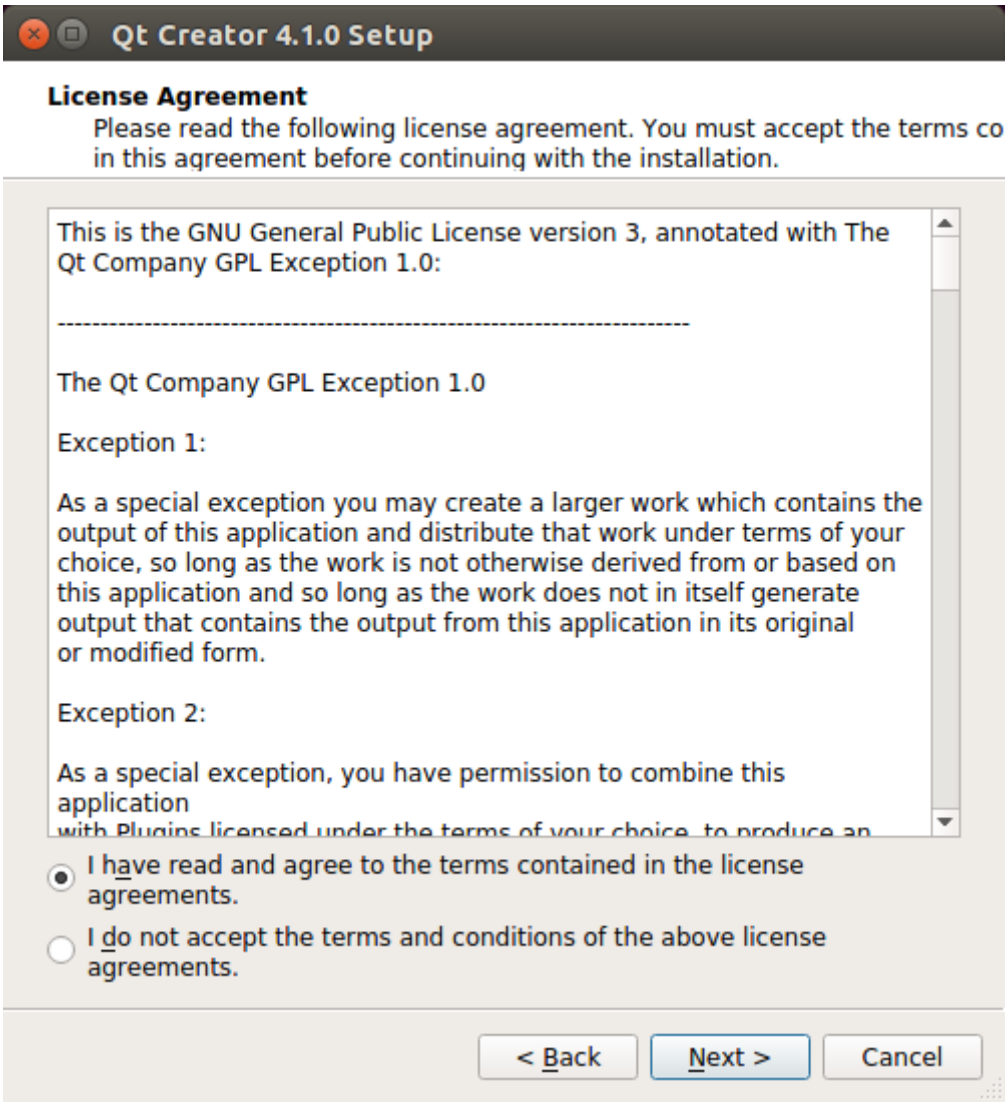
弹出图形窗口：



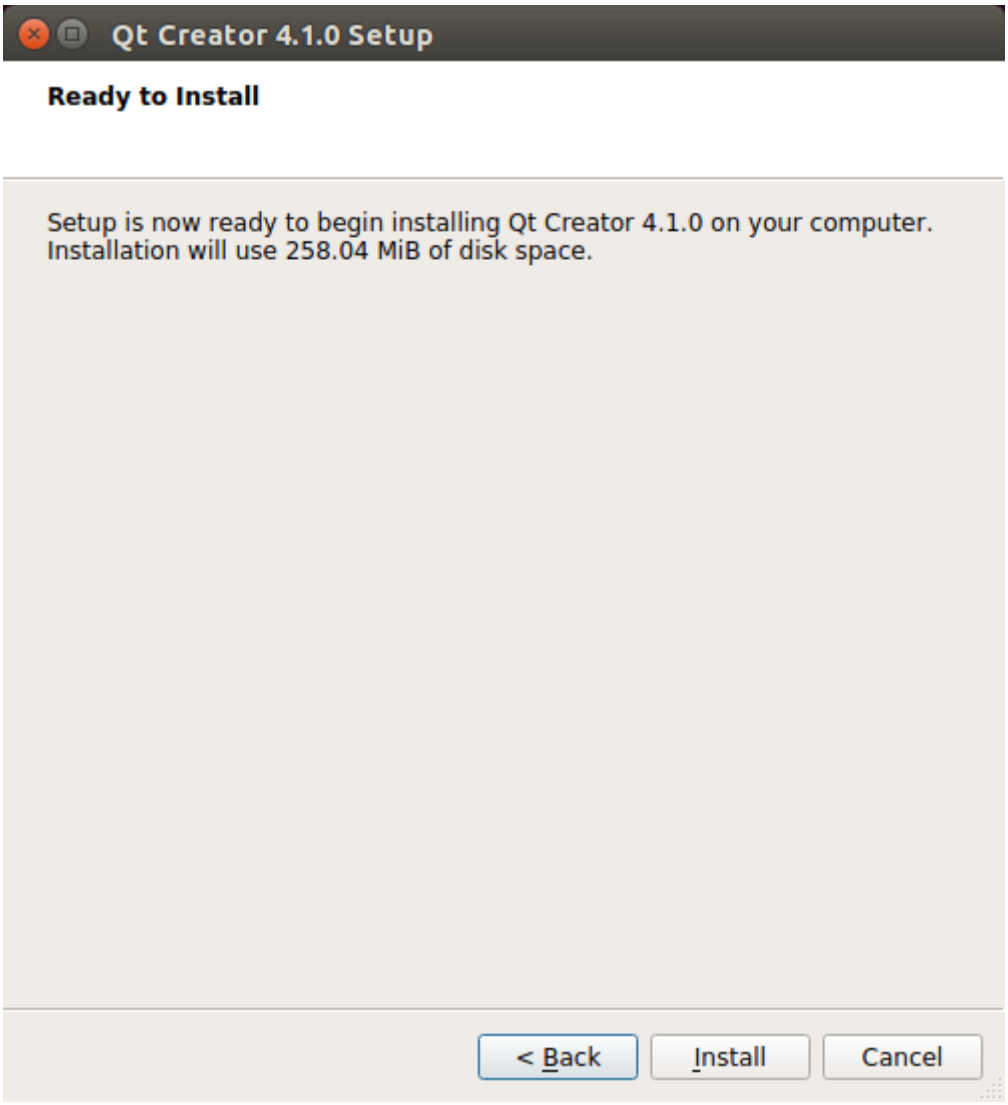

Qt Creator 4.1.0 Setup
Qt Account - Your unified login to everything Qt
Create Once.
Deploy Everywhere.
Please log in to Qt Account
Login
Email
Password
Forgot password?
Need a Qt Account?
Sign-up
Valid email address
Password
Confirm Password
I accept the service terms.
Settings
< Back
Skip
Cancel


Qt Creator 4.1.0 Setup
Setup - Qt Creator 4.1.0
Welcome to open source Qt Creator 4.1.0 setup.
Create Once.
Deploy Everywhere.
< Back
Next >
Quit

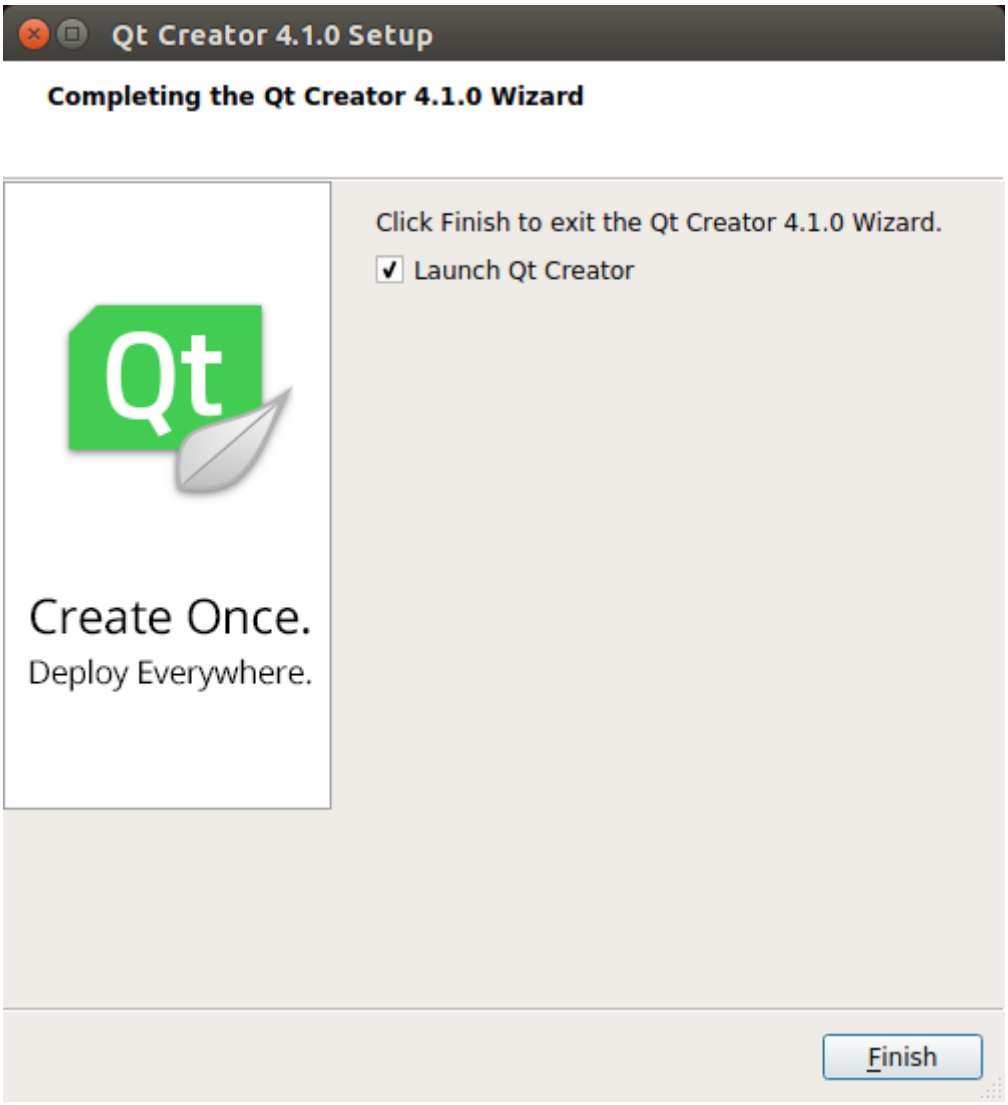

Qt Creator 4.1.0 Setup
Installation Folder
Please specify the folder where Qt Creator 4.1.0 will be installed.
/home/david/qtcreator-4.1.0
Browse...
< Back
Next >
Cancel


Qt Creator 4.1.0 Setup
License Agreement
Please read the following license agreement. You must accept the terms co
in this agreement before continuing with the installation.
This is the GNU General Public License version 3, annotated with The
Qt Company GPL Exception 1.0:

The Qt Company GPL Exception 1.0
Exception 1:
As a special exception you may create a larger work which contains the
output of this application and distribute that work under terms of your
choice, so long as the work is not otherwise derived from or based on
this application and so long as the work does not in itself generate
output that contains the output from this application in its original
or modified form.
Exception 2:
As a special exception, you have permission to combine this
application
I have read and agree to the terms contained in the license
agreements.
I do not accept the terms and conditions of the above license
agreements.
< Back
Next >
Cancel



根据指引信息，将 Qt creator 安装到$HOME/qtcreator-4.1.0
安装完成后如下图：



最后一步先不要启动 Qt Creator，需要从 TI SDK 中启动，因为有一些必要的环境变量需要设置。

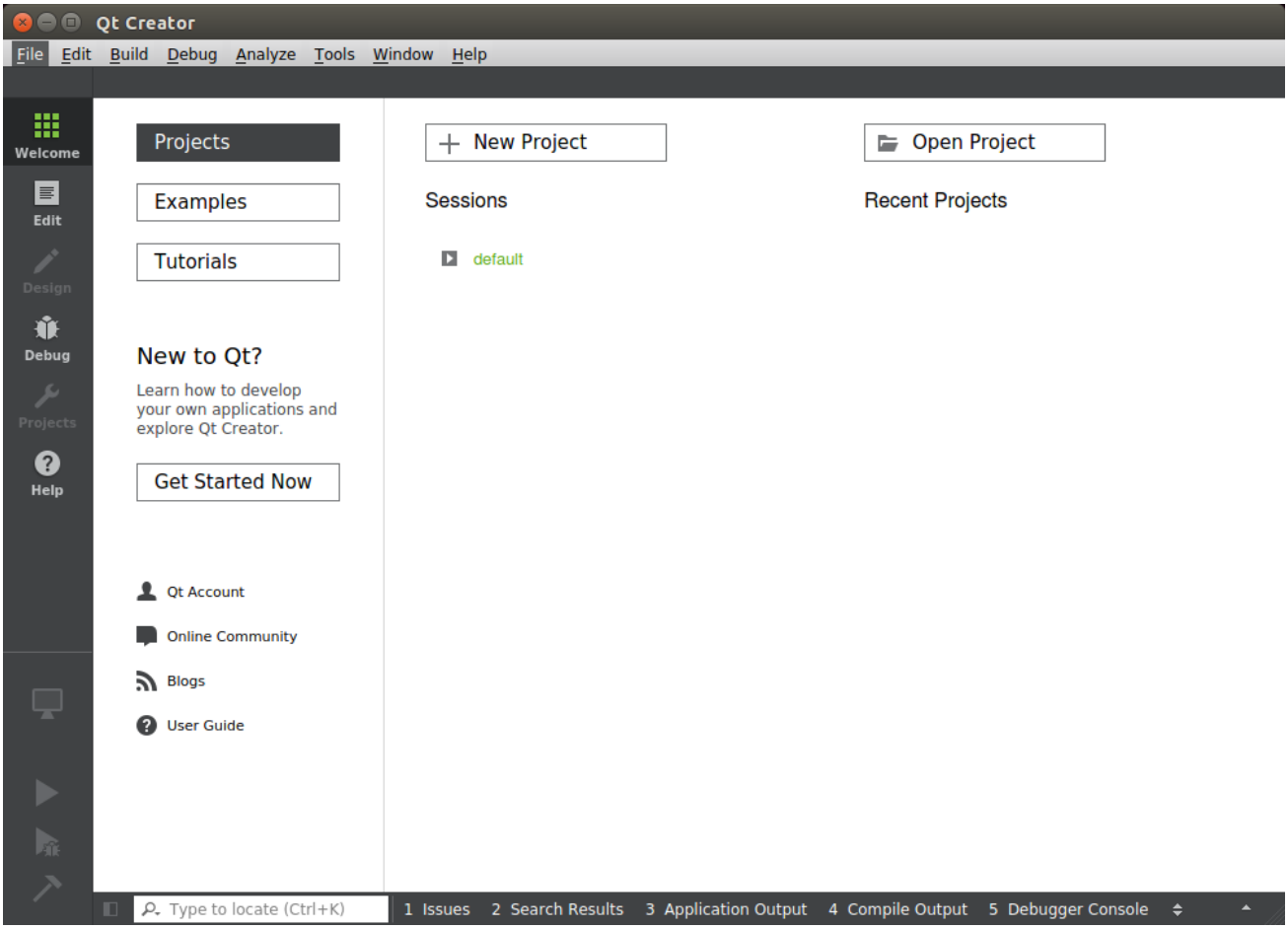
```
$ source /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/linux-devkit/environment-setup
```

然后切换到 Qt Creator 安装目录./qtcreator 启动

```
[linux-devkit]:~> cd ~/qtcreator-4.1.0/bin/
[linux-devkit]:~/qtcreator-4.1.0/bin> ./qtcreator
```

5.2.2 配置 Qt Creator

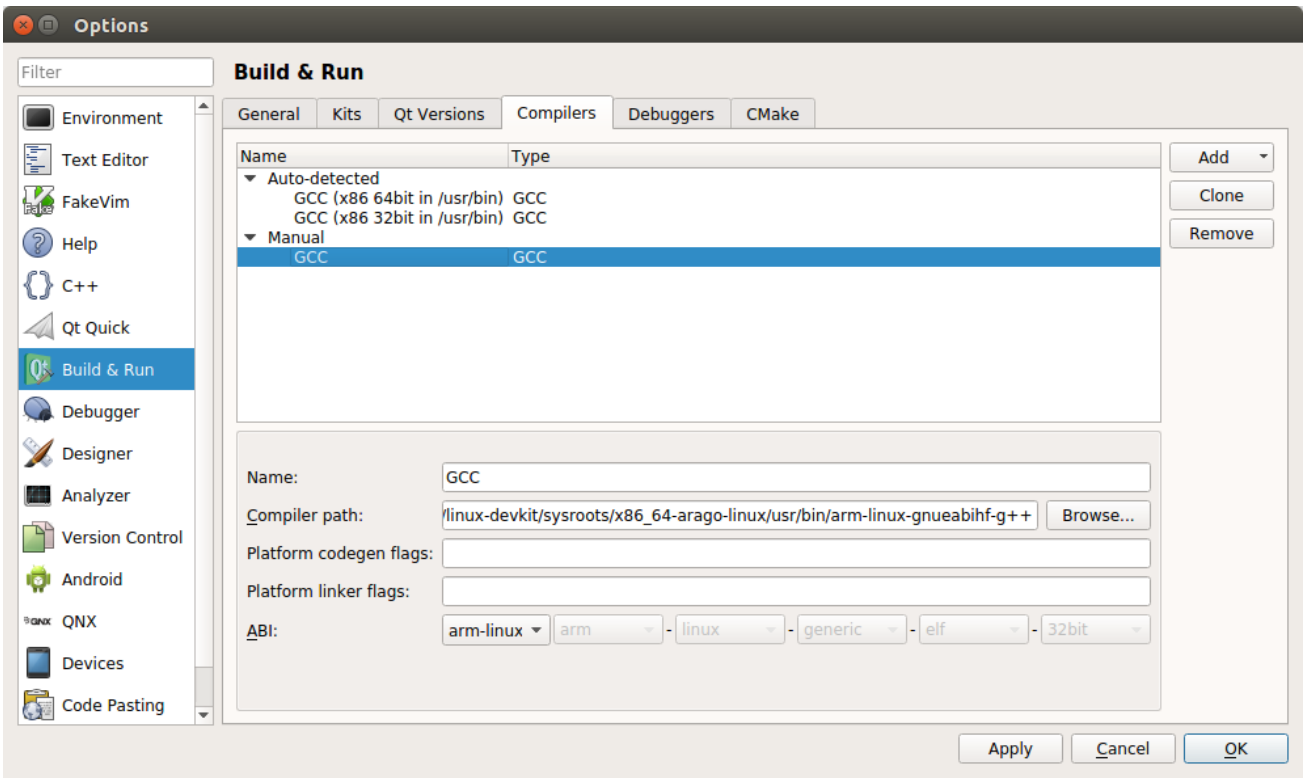
Qt Creator 启动后如下图所示：



在使用 Qt Creator 开发在 AM5728 上运行的程序前，需要配置交叉编译工具、QT 版本、Debuggers 版本、Kits。

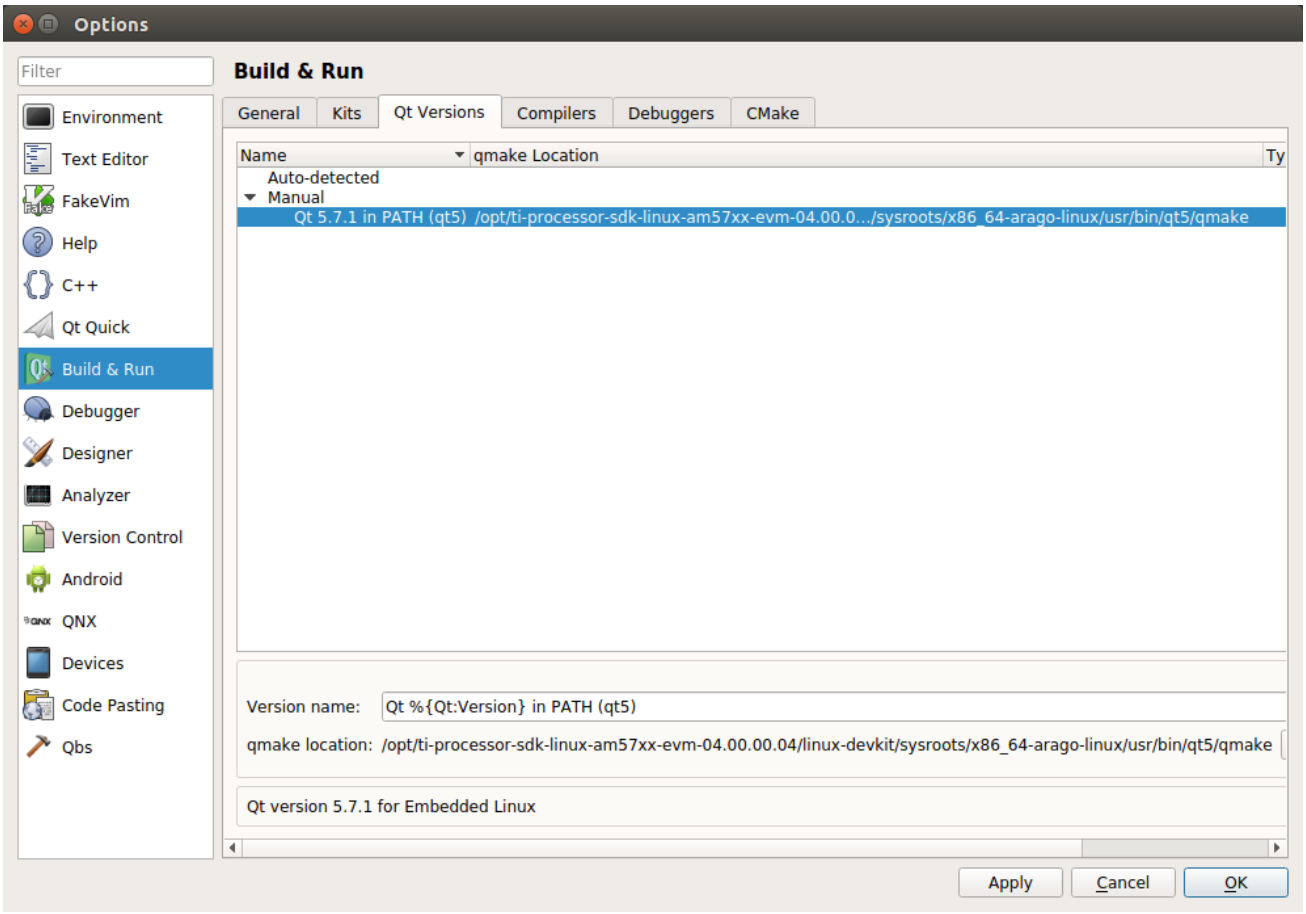
◆ 配置交叉编译工具链

点击菜单栏 Tool->Options->Build&Run->Compilers->Add->GCC，点击 Browse 选择 Compiler path 为 /opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnueabihf-g++(TI SDK 的 gcc 路径)，然后点击 Open，点击 Apply



◆ 配置 QT 版本

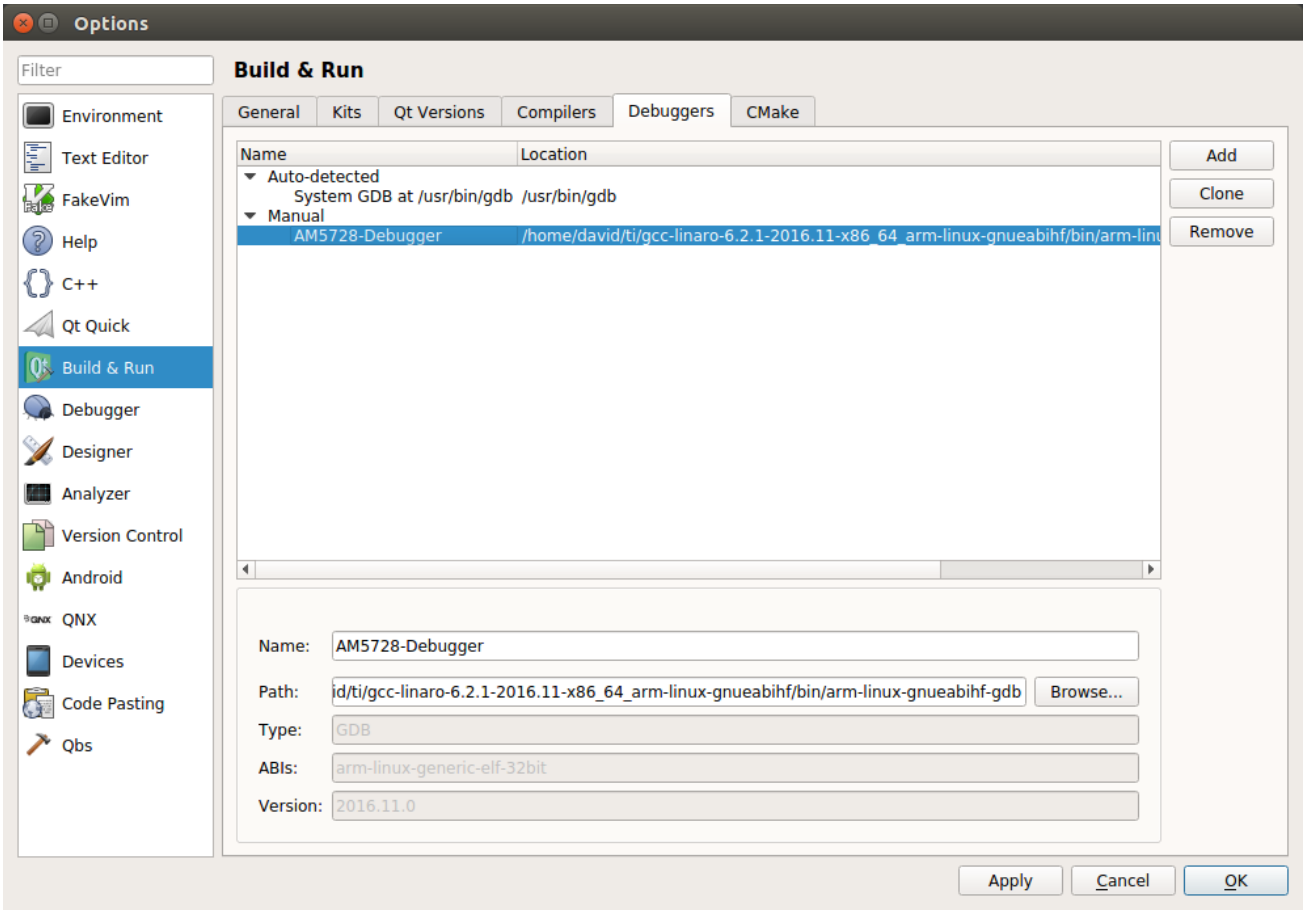
点击"Qt Versions->Add"，选择 TI SDK 的 qmake 文件路径：/opt/ti-processor-sdk-linux-am57xx-evm-04.00.00.04/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/qt5/qmake，然后点击 Apply



◆ 配置 Debuggers 版本

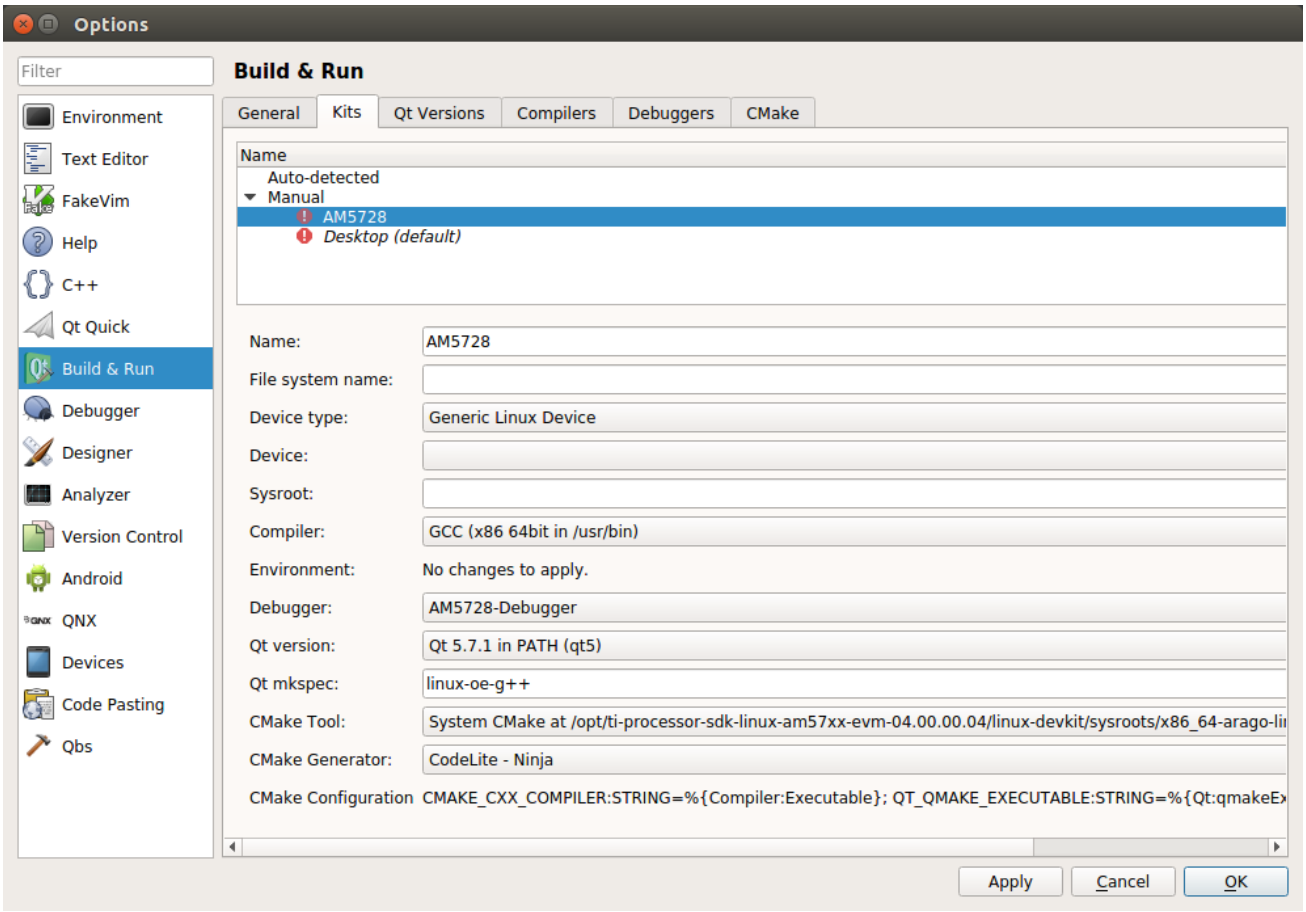
点击 Debuggers 选项，点击 Add，点击 Browse，选择交叉编译器安装目录下的 GDB 编译器，如：/home/david/ti/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gdb

更改 Name 选项，输入 AM5728-Debugger，点击 Apply，完成设置



◆ 配置 Kits

点击菜单栏"Tool->Options->Build & Run->Kits->Add"，更改 Name 为 AM5728，Device type 为 Generic Linux Device，在 Qt mkspec 选项中输入 linux-oe-g++，配置完后点击 Apply，点击 OK

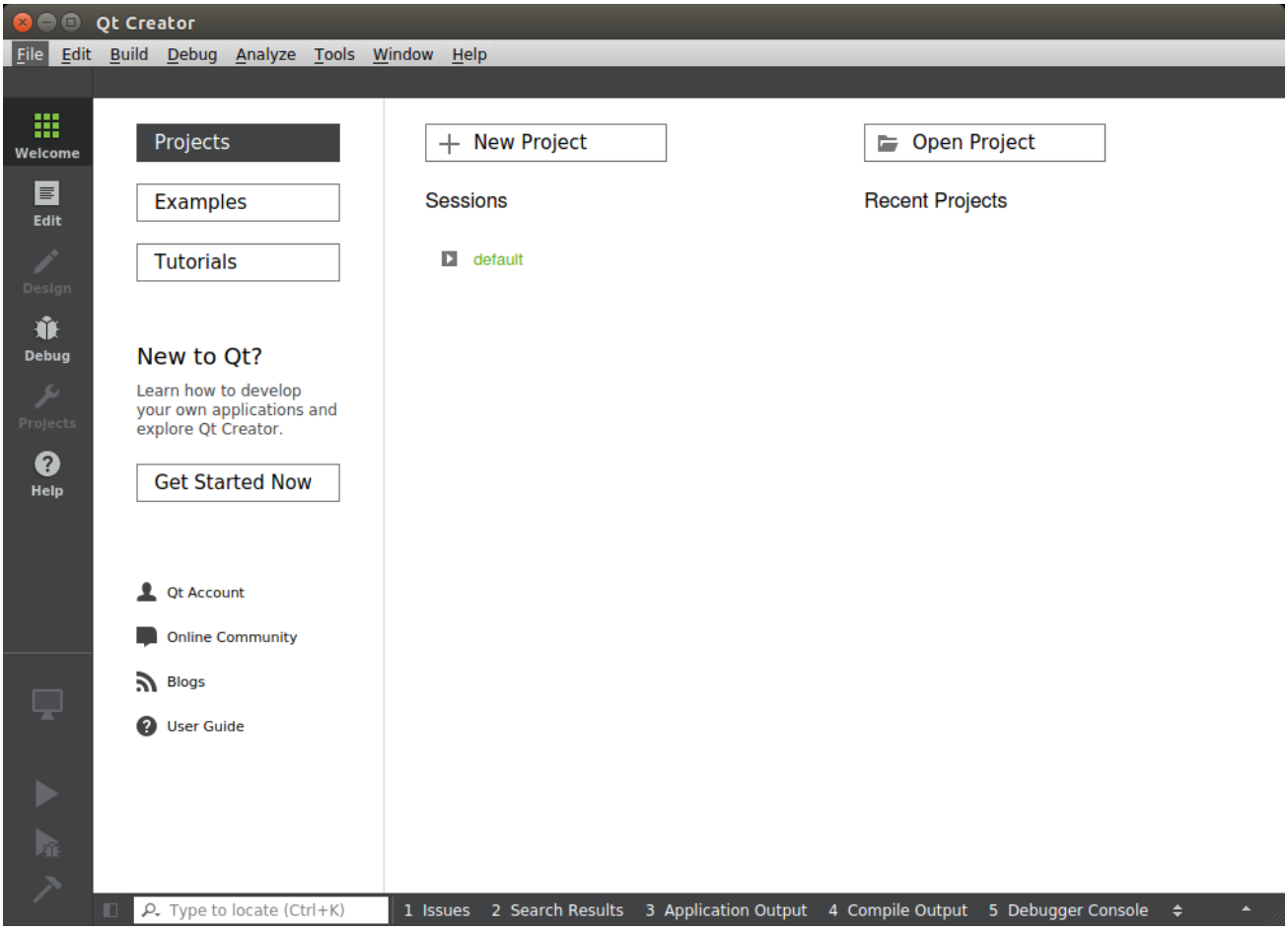


5.2.3 创建 demo

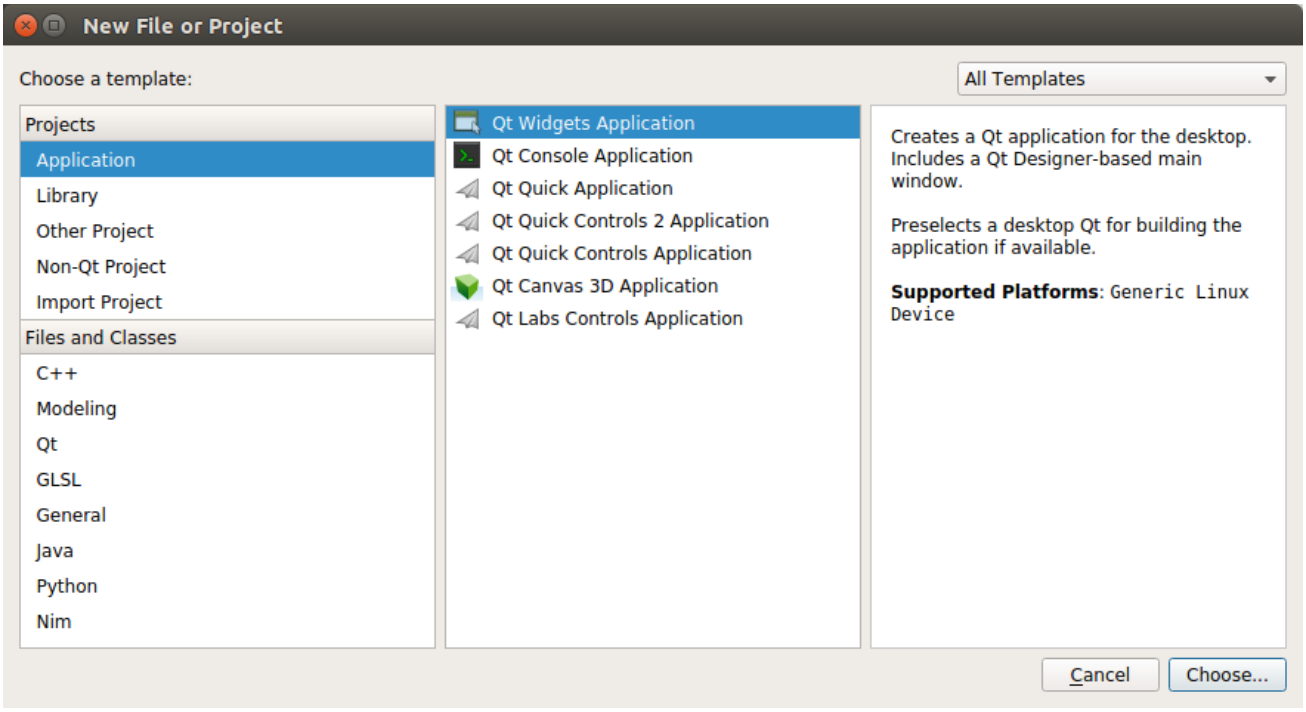
本章以一个简单的窗口程序 hello-world，介绍 Qt 界面的开发流程。

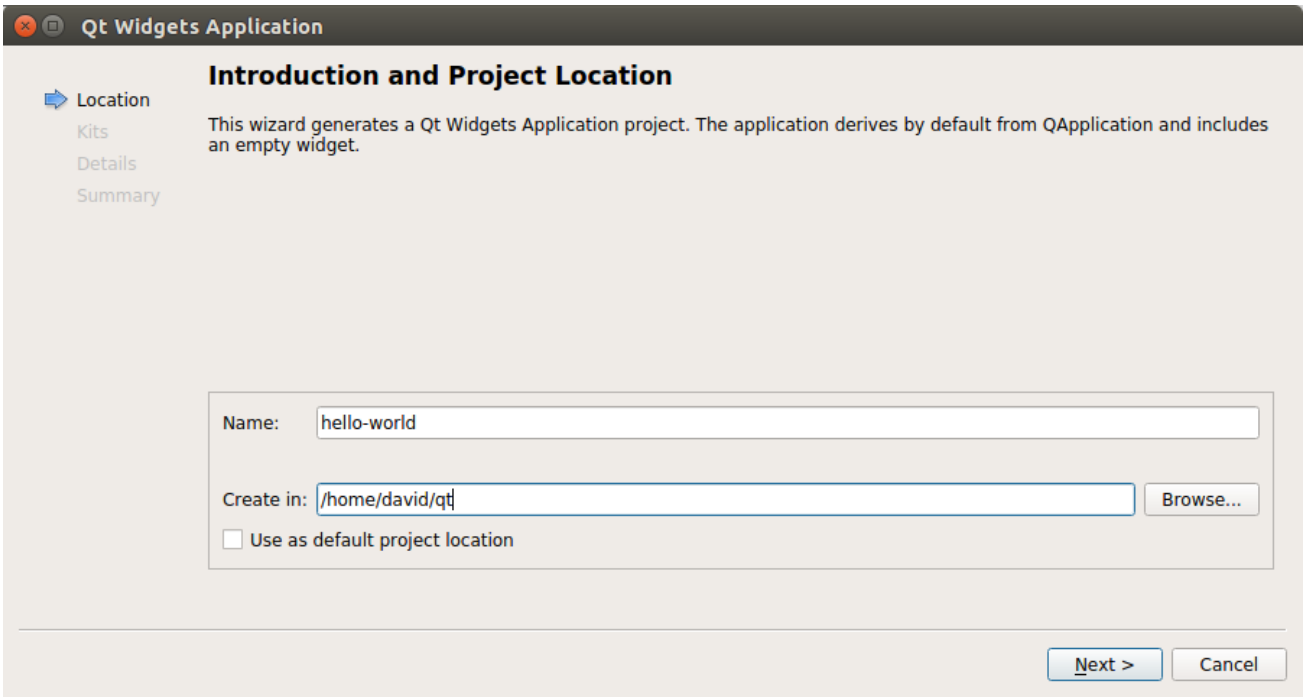
◆ 新建工程

同样，从 SDK 中启动 Qt Creator，源代码存放到/home/david/ti

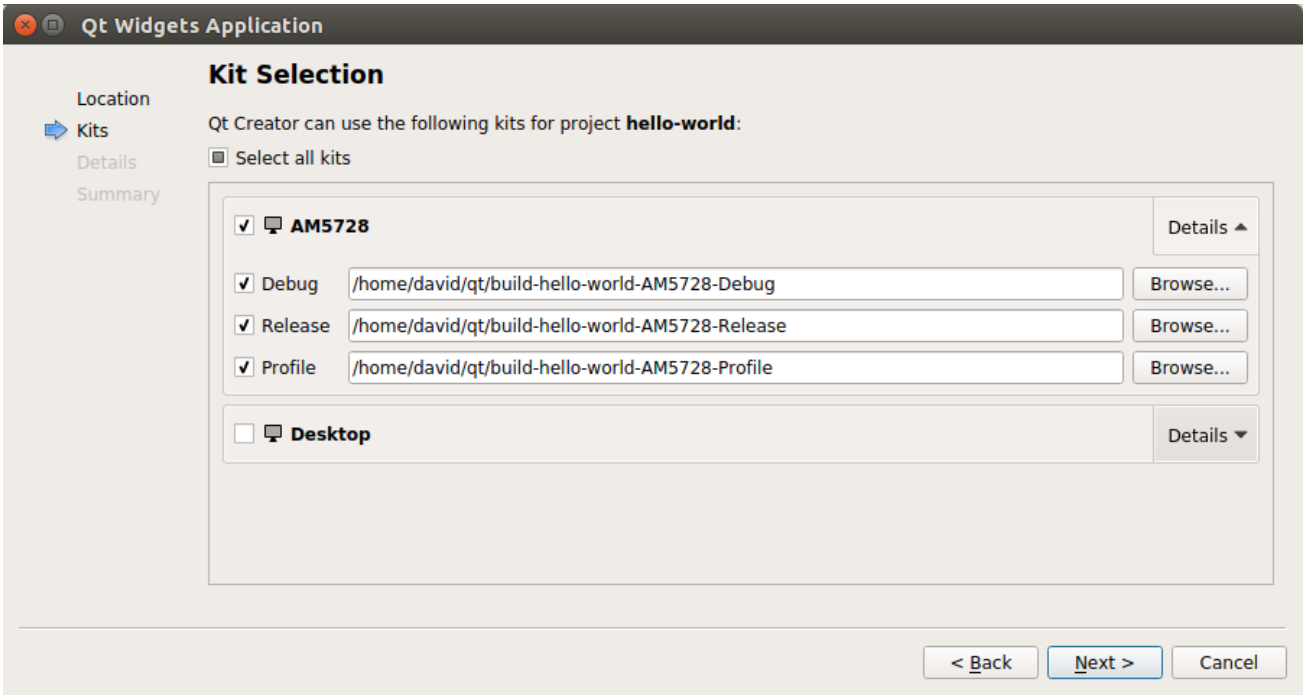


点击左上角 File-New File or Project 菜单

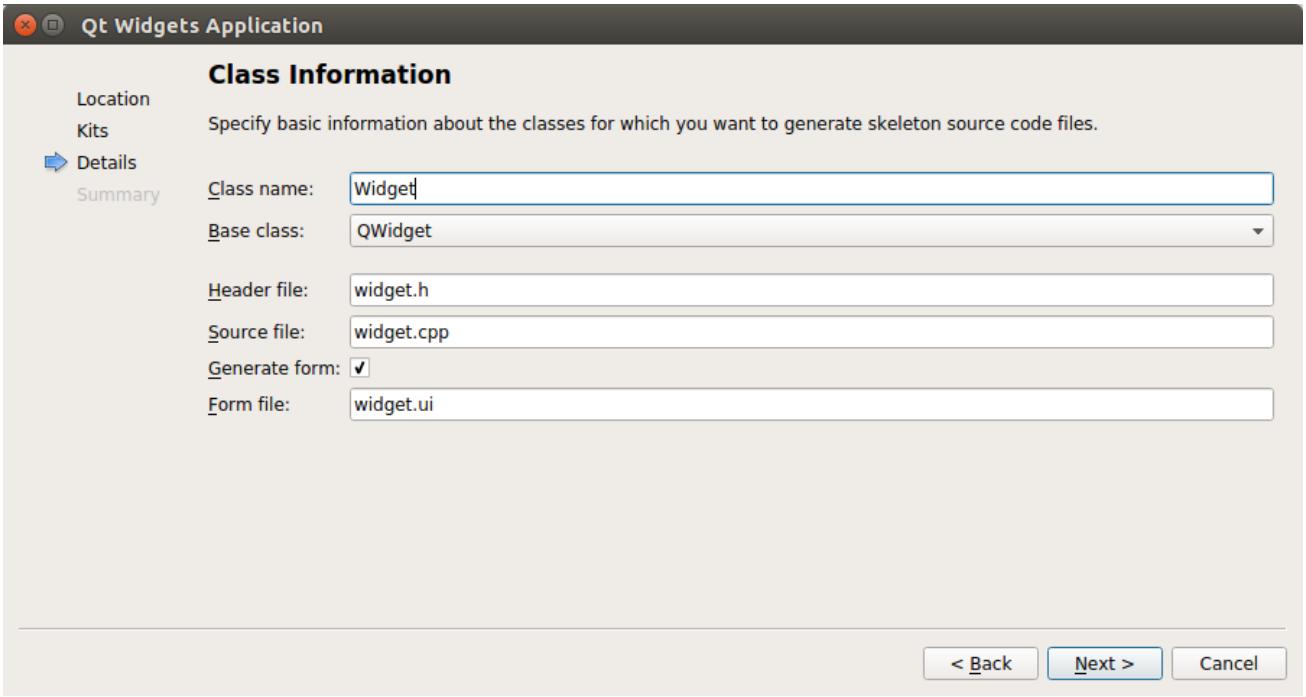




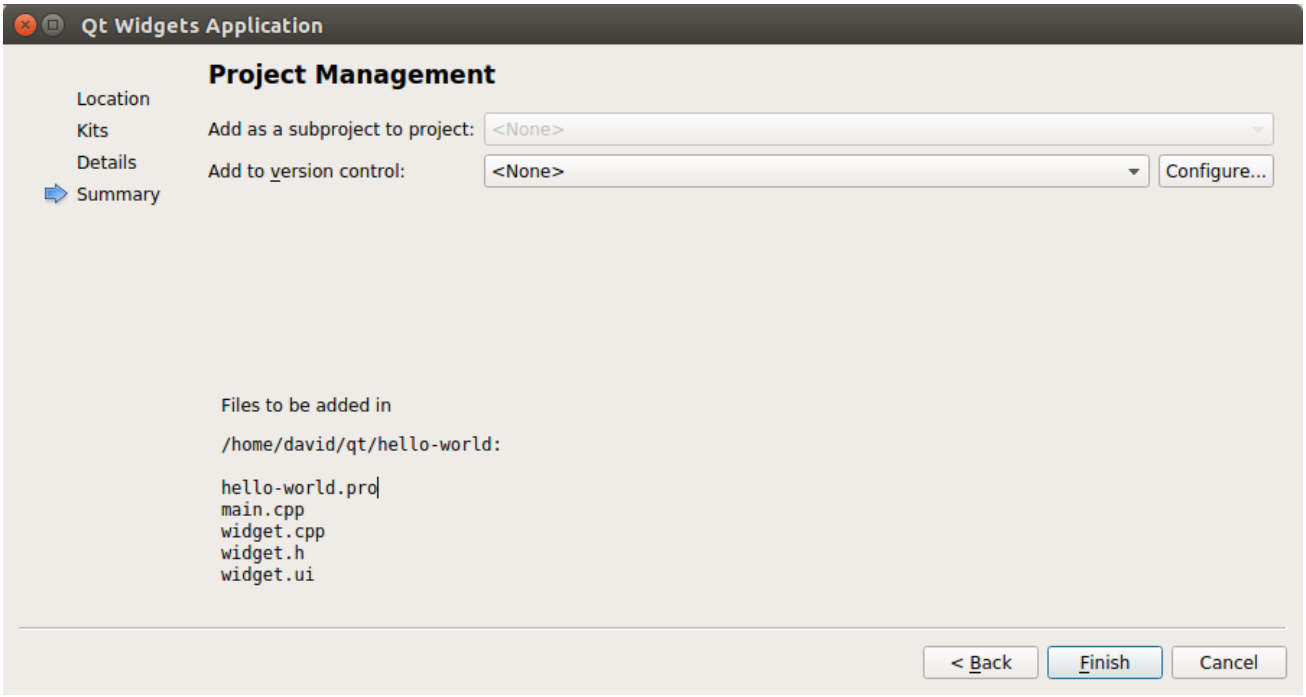
选择 Kit: AM5728



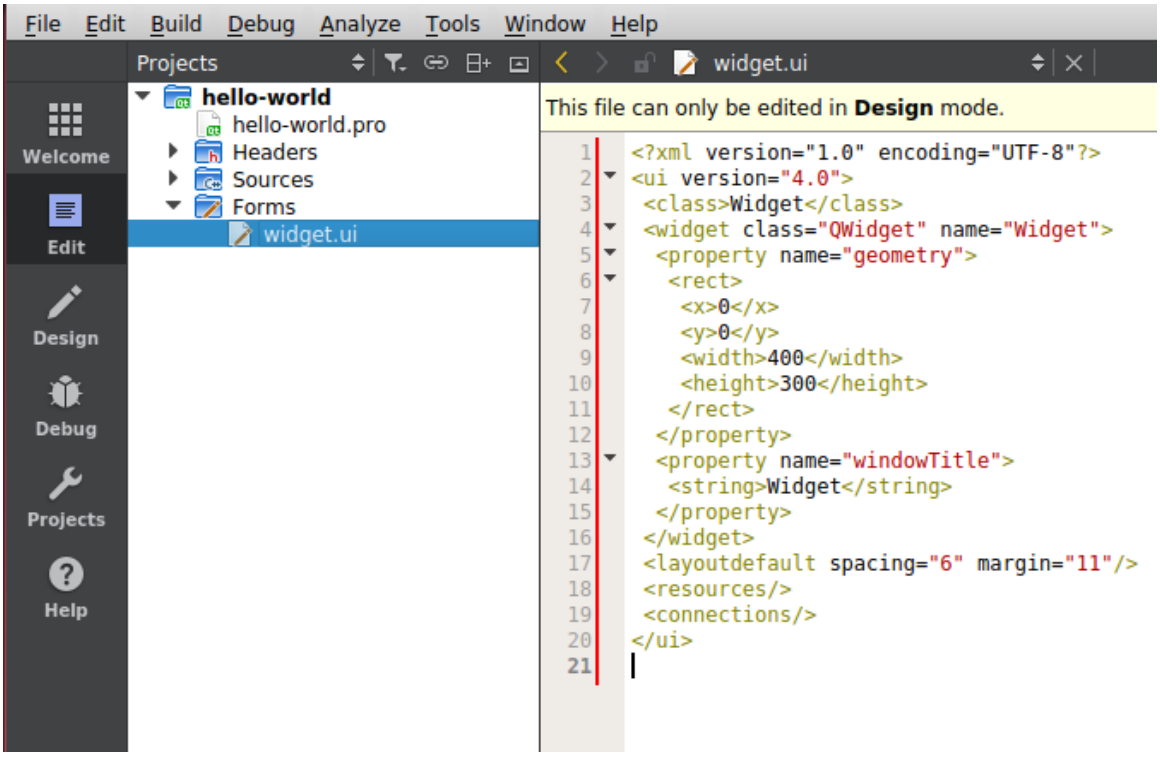
选择 Base Class: Qwidget



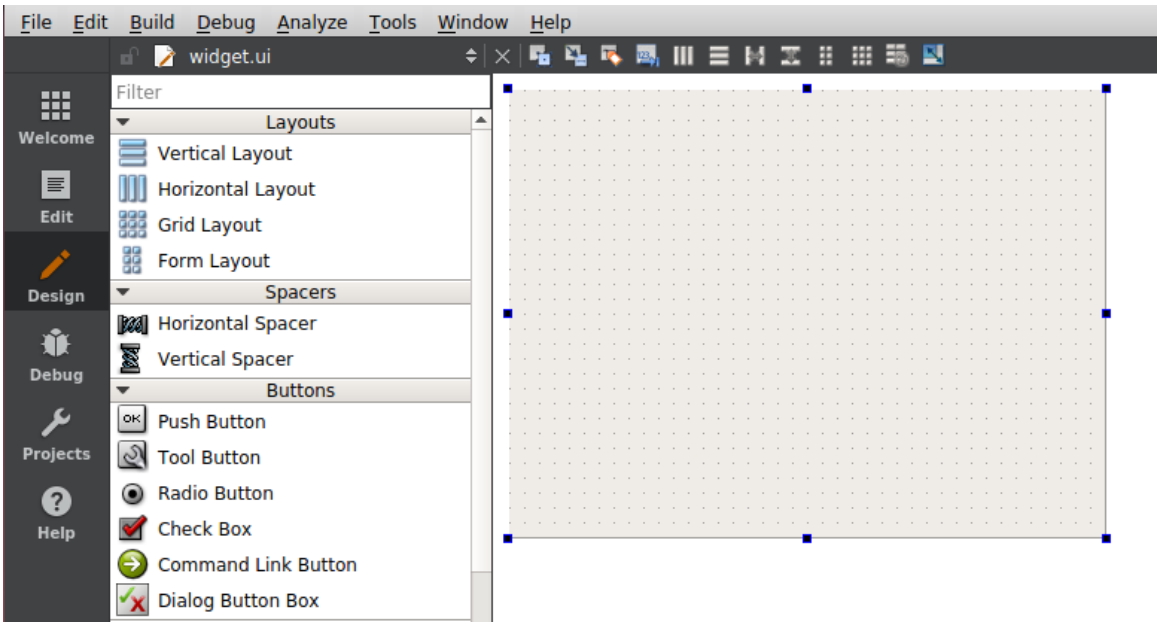
点击 Finish 进入到新建工程



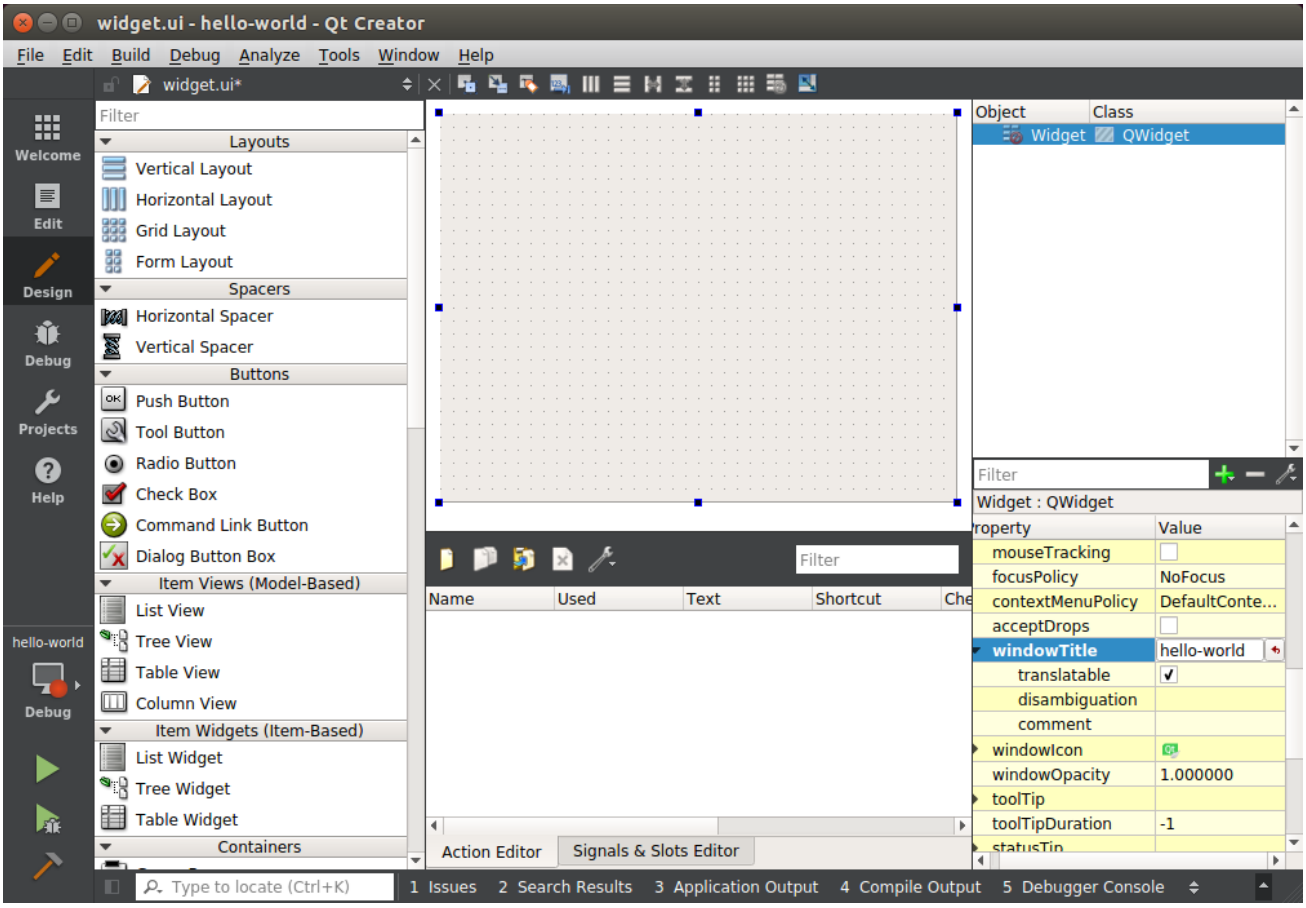
- 设计 UI 界面



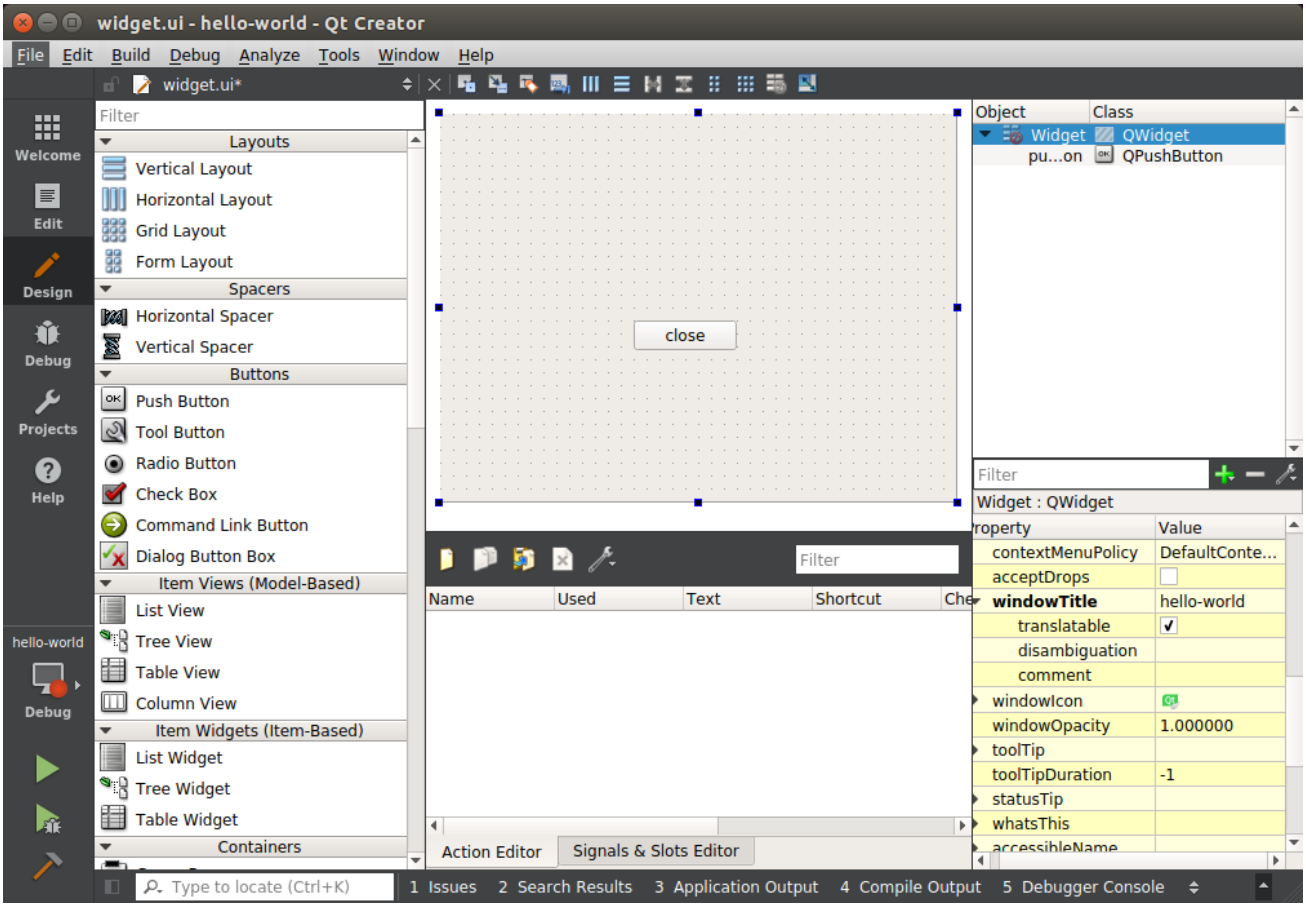
双击 widget.ui 进入 UI 设计器



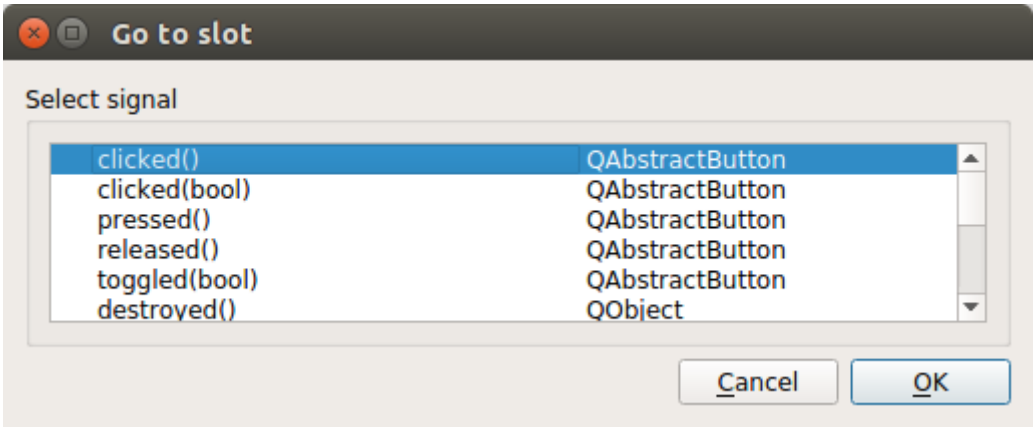
在右下角的 Property 属性窗口中，修改 windowTitle 字段，这里输入 hello-world



在左侧 Buttons 窗口拖动一个 Push Button 到 UI 上，双击修改名称为 close



选中 close 按钮，右键选择 Go to slot 进入到代码编辑，



◆ 编辑代码

编辑 on_pushButton_clicked 函数的响应代码，这里输入 close()，功能是关闭窗口。

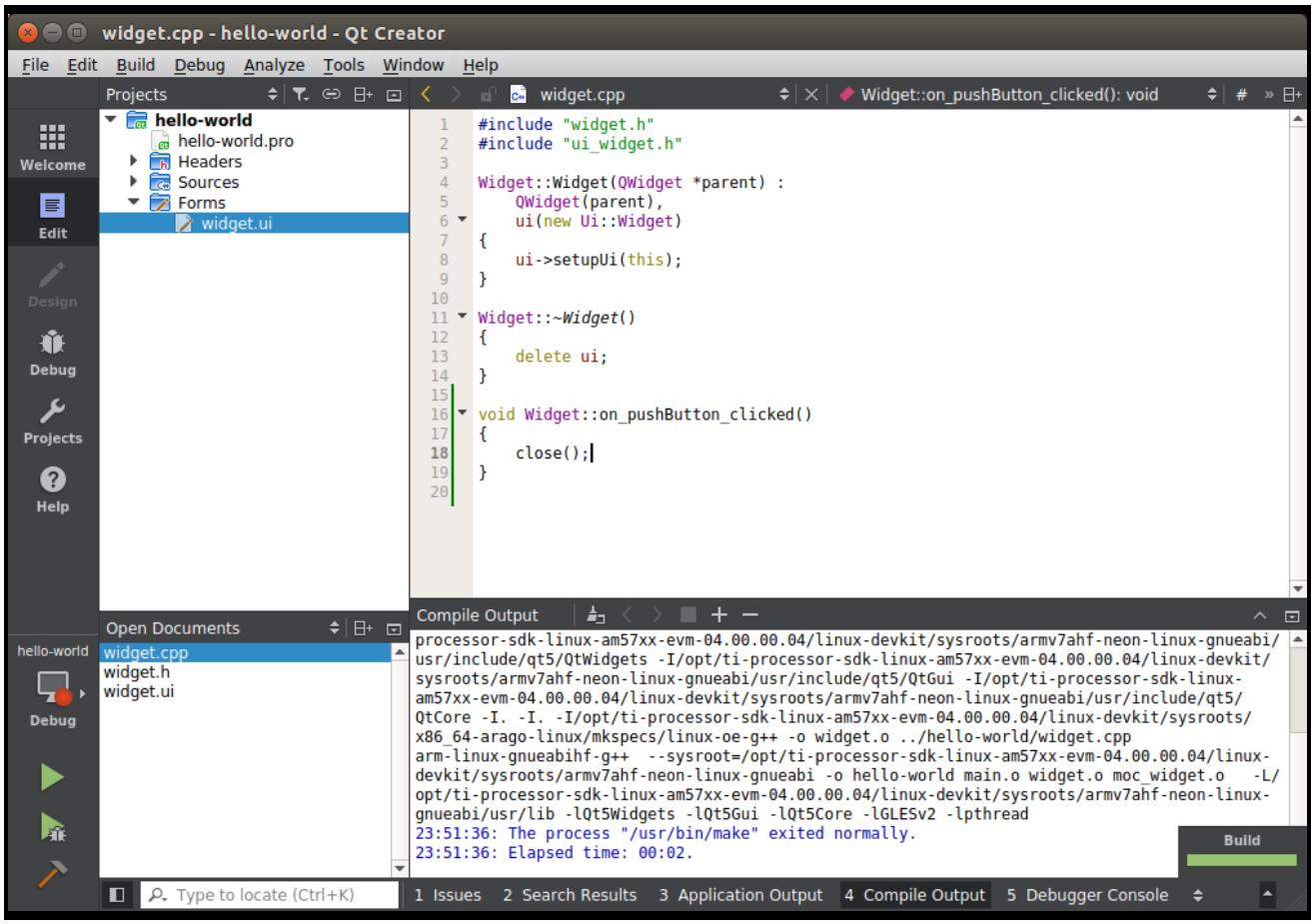
◆ 编译

点击左下角的锤子形状  的快捷按钮，编译工程，生产的可执行文件在

$HOME/qt/build-hello-world-AM5728-Debug 目录

通过 file 命令可以看到这是一个运行在 ARM 平台的可执行程序

```
$ file hello-world
hello-world: ELF 32-bit LSB  executable, ARM, EABI5 version 1 (GNU/Linux), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=cd9018247cd88be33eb2f59fb56fe7af7fee37ea, not stripped
```

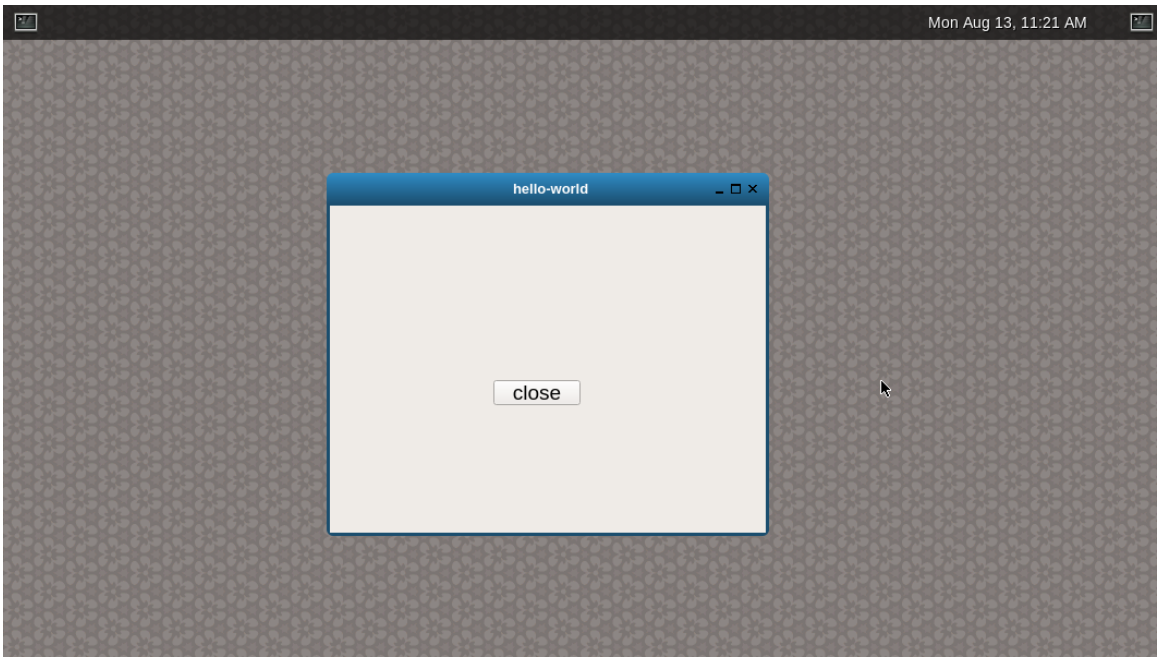


5.2.4 在 AM5728 板上运行

- 在 weston 环境运行

将生产的执行文件 hello-world 拷贝到 AM5728 板上，./hello-world 运行。

运行后的显示如下：

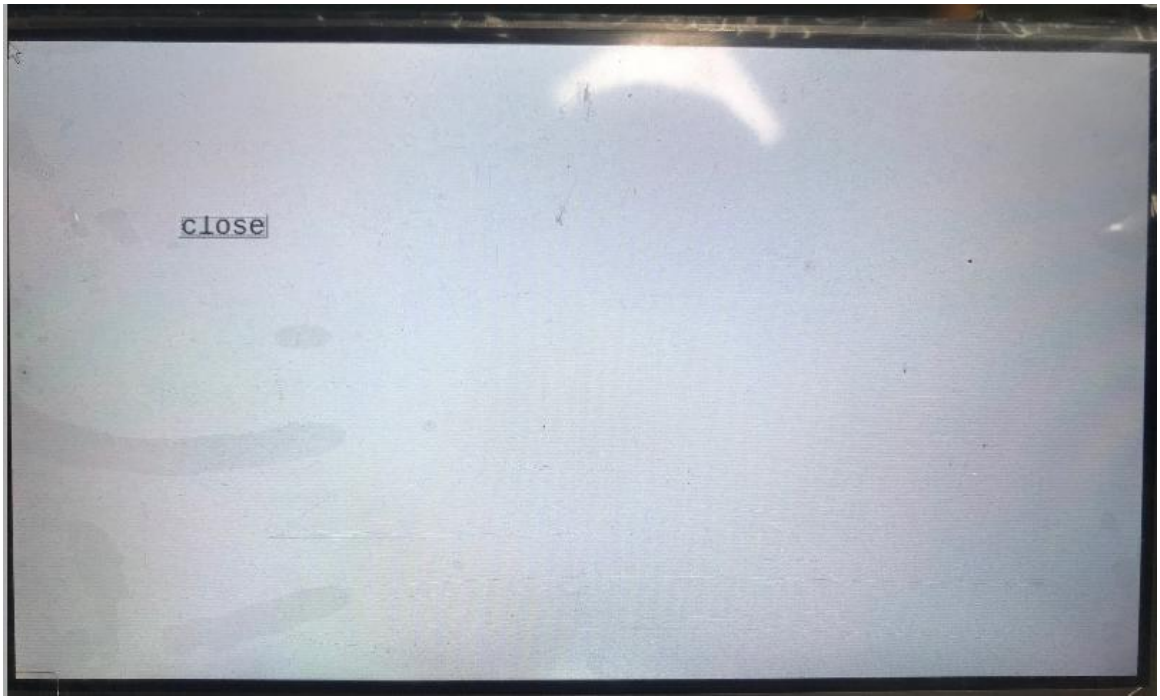


鼠标点击 close 按钮（或者触摸屏点击）即可关闭窗口。

- 脱离 Weston 运行

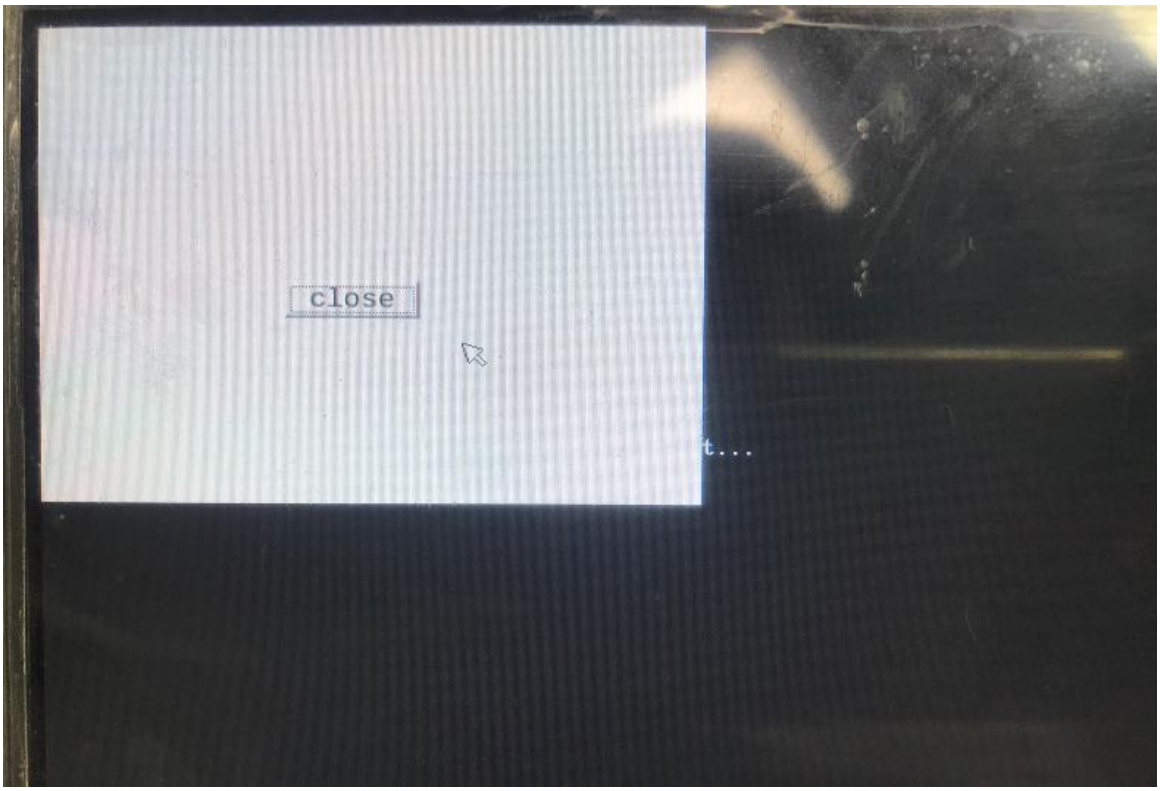
QT 程序也可以脱离 weston 环境运行，先关闭 weston

```
# /etc/init.d/weston stop
# ./hello-world -platform eglfs
```



或者

```
# ./hello-world -platform linuxfb
```



5.3 视屏采集 demo

视频采集功能上有预览，拍照，录像。运行 UI 界面中 Camera 应用（图标为 

），可以测试视频的预览和拍照。

- Capture：拍照快门，用于拍摄一张图片
- Switch：交换显示屏幕中的主显示和副显示窗口
- PIP：关闭副显示窗口
- Exit：退出应用

拍摄后的照片可以在 Gallery 中浏览（图标为 

），其界面大致如下，

Delete、Quit、Previous、Next 按钮分别是删除当前图片、退出应用、上一张图片、下一张图片。

视频录制即录像功能，由 gstreamer 来实现，命令行参考：

```
# gst-launch-1.0 -e v4l2src device=/dev/video2 num-buffers=1000 io-mode=5 ! 'video/x-raw, \
format=(string)NV12, width=(int)1920, height=(int)1080, framerate=(fraction)30/1' ! ducatimpeg4enc
bitrate=12000 ! \
queue !mpeg4videoparse !qtmux !filesink location=out.mp4
```

播放上面录制的文件，用下面的命令(假设生成的 mp4 文件全路径是：/media/out.mp4)：

```
# gst-launch-1.0playbinuri=file:///media/out.mp4 video-sink=kmssink
```

5.4 双屏显示 demo

双显示的一个 Demo，可以演示 OpenGL、双摄像头、双显示同时工作。此应用通过 OpenGL 构造一个旋转正方体，通过 V4L2 视频采集接口获取双摄像头数据，然后分别显示在 LCD 和 HDMI 显示屏幕上。用来演示平台强大的视频处理和显示能力。前提需要关闭 Weston

```
#/etc/init.d/weston stop
```

具体的测试命令如下：

- 双屏分别显示旋转立方体和一个摄像头

```
# kmscube-camera -a
```

- 双屏分别显示旋转立方体和 #0 摄像头

```
#kmscube-camera -a -i 0
```

- 双屏分别显示旋转立方体和 #1 摄像头

```
#kmscube-camera -a -i 1
```

- 单屏显示旋转立方体

```
#kmscube-camera -c 32
# kmscube-camera -c 36
```

- 双屏显示，一个屏幕显示旋转立方体，一个屏幕交替显示两个摄像头

```
#kmscube-camera -a -i 2
```

第6章 基于 Yocto 的根文件系统构建

Yocto 可以构建出完整的嵌入式系统镜像，本文侧重于构建根文件系统。更多关于 Yocto 的信息请访问官方网站 https://www.yoctoproject.org/docs/2.5/mega-manual/mega-manual.html

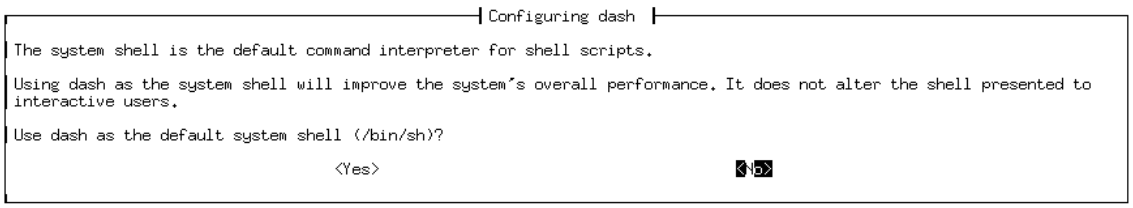
搭建 Yocot 构建环境需要 PC 机有较好的硬件性能和足够的内存和硬件空间，并且需要充裕的网络带宽，推荐至少 200G 硬件空间和 8G 内存。

6.1 安装需要的工具软件

```
$ sudo apt-get install git build-essential python diffstattexinfo gawk chrpath dos2unix wget unzip socatdoxygen
libc6:i386 libncurses5:i386 libstdc++6:i386 libz1:i386
```

6.2 配置 bash

```
$ sudo dpkg-reconfigure dash
```



选择”no”

6.3 安装编译器

如果此前已经安装好了交叉编译工具链，此处可以忽略

```
$ wget
https://releases.linaro.org/components/toolchain/binaries/6.2-2016.11/arm-linux-gnueabihf/gcc-linaro-6.2.1-
2016.11-x86_64_arm-linux-gnueabihf.tar.xz
$ tar -Jxvf gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz -C $HOME
```

6.4 获取 oe-layertool-setup.sh

```
$ git clone git://arago-project.org/git/projects/oe-layersetup.git tisdk
$ cd tisdk
$ ./oe-layertool-setup.sh -f configs/processor-sdk/processor-sdk-04.00.00.04-config.txt
```

6.5 bitbake 构建

```
$ cd ..
$ cd build
$ . conf/setenv
$ export PATH=$HOME/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin:$PATH
$ MACHINE=am57xx-evm bitbake arago-core-tisdk-image
```

初次构建需要花费较长时间，大约几小时到几十小时不等，具体时间因 PC 性能和网络带宽而异。

构建完成后，生成的文件系统在

tisdk/build/arago-tmp-external-linaro-toolchain/deploy/images/am57xx-evm/tisdk-rootfs-image-am57xx-evm.tar.xz

第7章 附录

7.1 硬件

详细硬件介绍请参考：

核心板：《EM-TF-SOM-AM5728 硬件用户手册》

底板：《EM-TF-BB-AM5728 硬件用户手册》

第8章 技术支持和保修服务

8.1.1 技术支持

英蓓特科技对所销售的产品提供一年的免费技术支持服务，技术支持服务范围：

- 提供英蓓特科技嵌入式平台产品的软硬件资源；
- 帮助用户正确地编译和运行我们提供的源代码；
- 用户在按照本公司提供的产品文档操作的情况下，如本公司的嵌入式软硬件产品出现异常问题，我们将提供技术支持；
- 帮助用户判定是否存在产品故障。
- 以下情况不在我们的免费技术支持服务范围内，但我们将根据情况酌情处理：
  - 用户自行开发中遇到的软硬件问题；
  - 用户自行修改嵌入式操作系统遇到的问题；
  - 用户自己的应用程序遇到的问题；
  - 用户自行修改本公司提供的软件代码遇到的问题。

8.1.2 保修服务

- 产品自出售之日起，在正常使用状况下为印刷电路板提供 12 个月的免费保修服务；
- 以下情况不属于免费服务范围，英蓓特科技将酌情收取服务费用：
  - 无法提供产品有效购买凭证、产品识别标签撕毁或无法辨认，涂改标签或标签与实际产品不符；
  - 未按用户手册操作导致产品损坏的；
  - 因天灾 (水灾、火灾、地震、雷击、台风等) 或零件之自然耗损或遇不可抗拒力导致的产品外观及功能损坏；
  - 因供电、磕碰、房屋漏水、动物、潮湿、杂 / 异物进入板内等原因导致的产品外观及功能损坏；
  - 用户擅自拆焊零件或修改而导致不良或授权非英蓓特科技认可的人员及机构进行产品的拆装、维修，变更产品出厂规格及配置或扩充非英蓓特科技公司销售或认可的配件及由此引致的产品外观及功能损坏；
  - 用户自行安装软件、系统或软件设定不当或由电脑病毒等造成的故障；
  - 非经授权渠道购得此产品者。
  - 非英蓓特科技对用户做出的超出保修服务范围的承诺（包括口头及书面等）由承诺方负责兑现，英蓓特科技恕不承担任何责任；
- 保修期内由用户发到我们公司的运费由用户承担，由我们公司发给用户的运费由我们承担；保修期外的全部运输费用由用户承担。
- 若板卡需要维修，请联系技术支持服务部。

英蓓特科技公司对于未经本公司许可私自寄回的产品不承担任何责任。

第9章 联系方式

- 电话：+86-755-33190846/33190847/33190848
- 邮箱：
  - 技术支持：support@embest-tech.com
  - 销售：chinasales@embest-tech.com
- 传真：+86-755-25616057
- 网站：http://www.embest-tech.cn
- 地址：深圳市南山区留仙大道 4093 号南山云谷创新产业园山水楼 4 楼 B