GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/4/23/33 32-bit MCU

应用笔记

AN033

目录

图索引

表索引

1. 简介

我们通常利用windows环境下的集成开发环境来开发单片机程序，例如keil或者IAR。本文介绍使在linux环境下用多个makefile管理RTOS工程的方法，在RTOS任务中实现一个LED闪烁功能。该方法可指定模块或文件进行编译。

2. 开发环境介绍

开发环境准备：

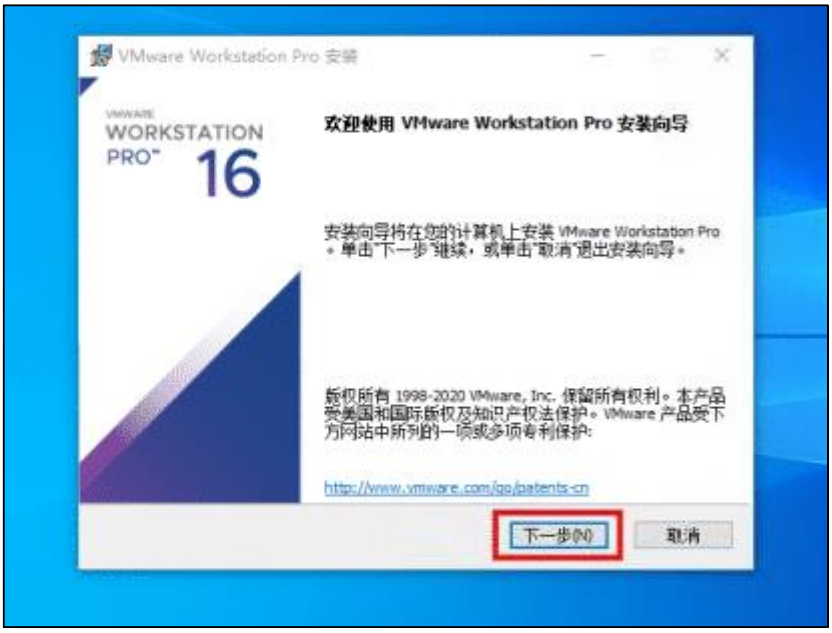
- 硬件平台：GD32F303-Test-V1.1
- 编译环境：ubuntu16.04
- 工具链： gcc-arm-none-eabi, gcc-arm-none-objcopy
- 烧录工具：SEGGR J-FlashVV6.50b

2.1. 安装 ubuntu 虚拟机

虚拟机软件下载地址：https://www.vmware.com/cn/products/workstation-pro/workstation-pro-evaluation.html.

双击运行安装包文件,根据安装向导，如*图 2-1. ubuntu 虚拟机安装向导 1* 点击下一步，选择默认的设置。

图 2-1. ubuntu 虚拟机安装向导 1



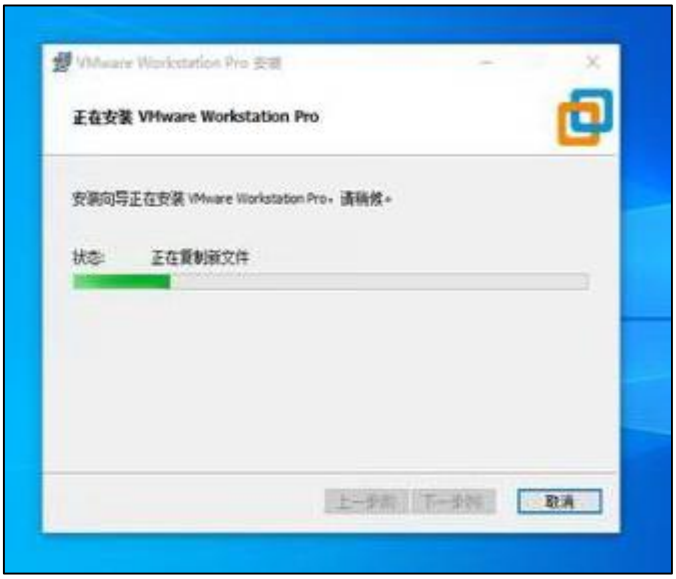
欢迎使用 VMware Workstation Pro 安装向导

安装向导将在您的计算机上安装 VMware Workstation Pro。单击"下一步"继续，或单击"取消"退出安装向导。

版权所有 1998-2020 VMware, Inc. 保留所有权利。本产品受美国和国际版权及知识产权法保护。VMware 产品受下方网站中所列的一项或多项专利保护:

http://www.vmware.com/go/patents-cn

图 2-2. ubuntu 虚拟机安装向导 2



VMware Workstation Pro 安装

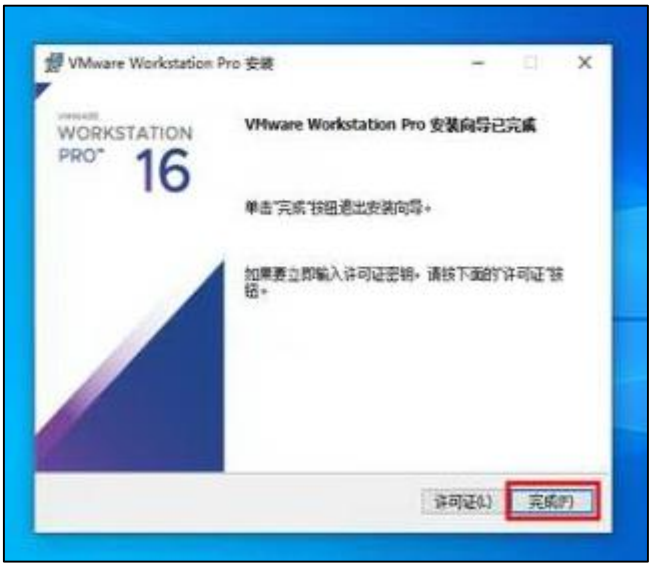
正在安装 VMware Workstation Pro

安装向导正在安装 VMware Workstation Pro。请稍候。

状态: 正在复制新文件

点击“完成”，完成安装.。

图 2-3. ubuntu 虚拟机安装向导 3



VMware Workstation Pro 安装

VMware Workstation Pro 安装向导已完成

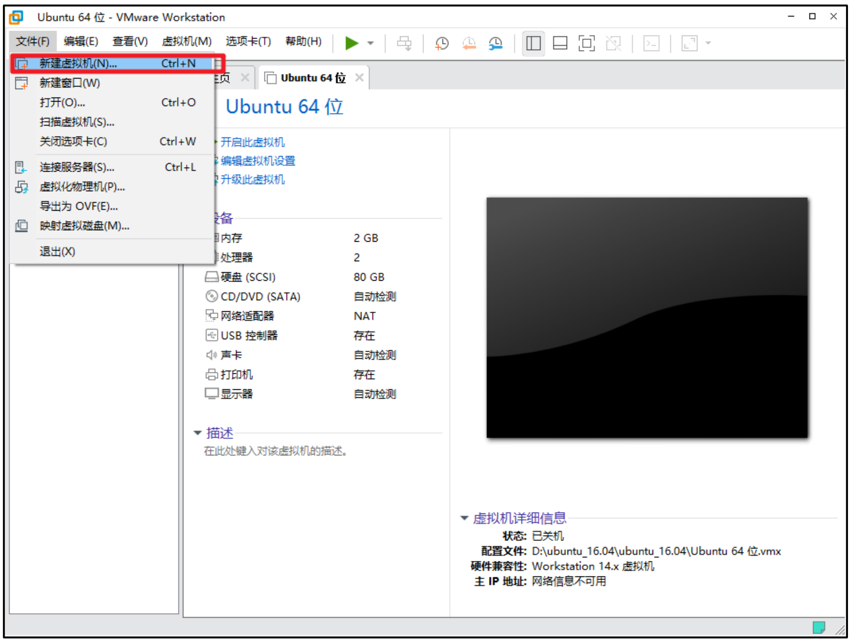
单击"完成"按钮退出安装向导。

如果要立即输入许可证密钥，请单击下面的"许可证"按钮。

在 VMware 中安装 ubuntu，下载 ubuntu 镜像文件，下载地址：http://mirrors.aliyun.com/ubuntu-releases/16.04/

Ubuntu 镜像下载完成后，打开虚拟机，点击“文件 --> 新建虚拟机”。

图 2-4. ubuntu 虚拟机安装向导 4

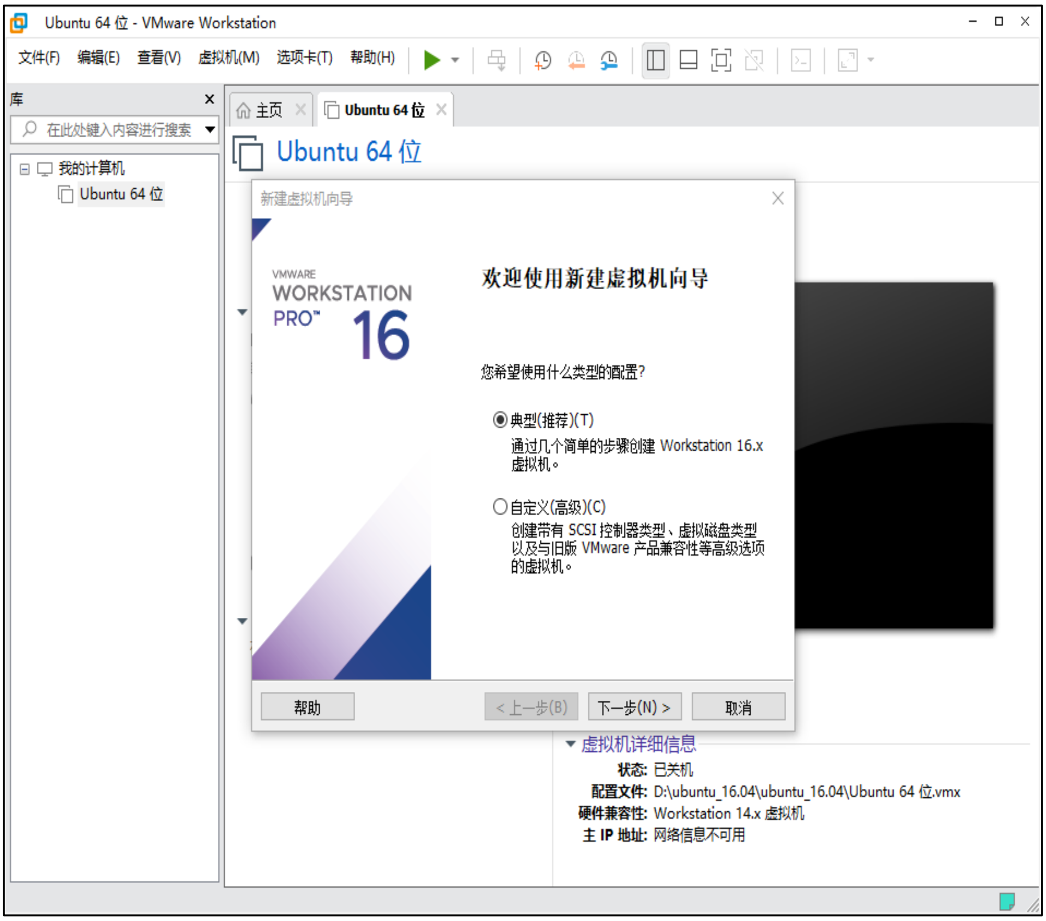


内存 2 GB
处理器 2
硬盘 (SCSI) 80 GB
CD/DVD (SATA) 自动检测
网络适配器 NAT
USB 控制器 存在
声卡 自动检测
打印机 存在
显示器 自动检测

状态: 已关机
配置文件: D:\ubuntu_16.04\ubuntu_16.04\Ubuntu 64 位.vmx
硬件兼容性: Workstation 14.x 虚拟机
主 IP 地址: 网络信息不可用

使用默认设置，点击“下一步”。

图 2-5. ubuntu 虚拟机安装向导 5



欢迎使用新建虚拟机向导

您希望使用什么类型的配置?

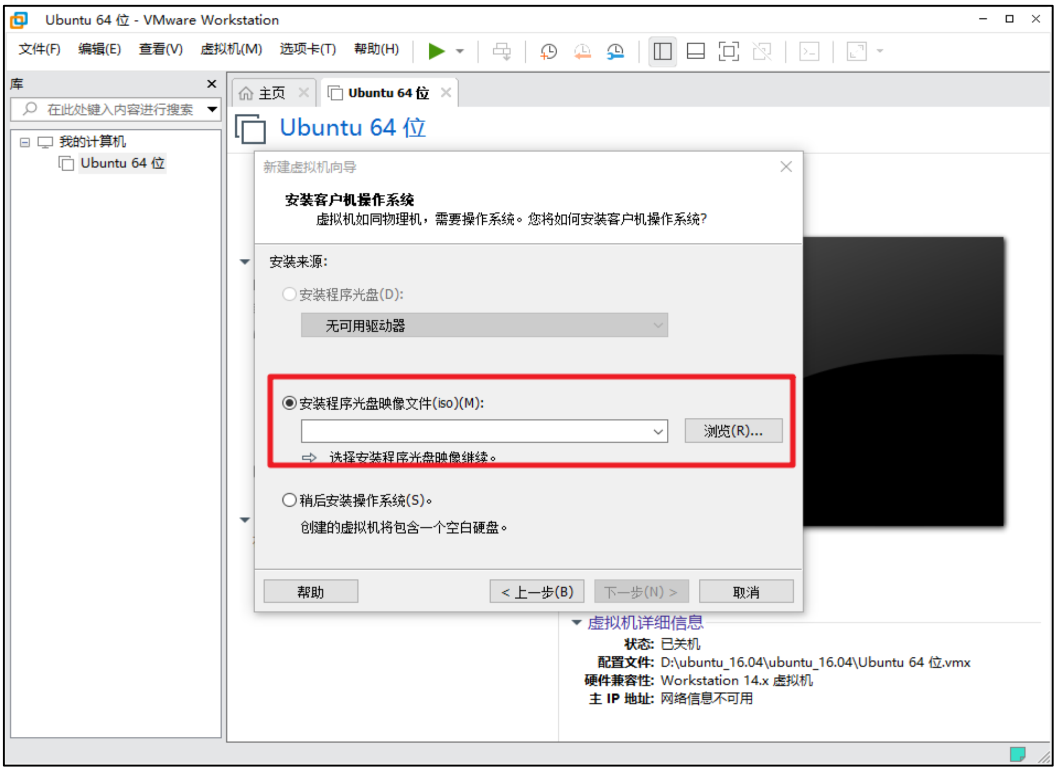
● 典型(推荐)(T)
通过几个简单的步骤创建 Workstation 16.x 虚拟机。

○ 自定义(高级)(C)
创建带有 SCSI 控制器类型、虚拟磁盘类型以及与旧版 VMware 产品兼容性等高级选项的虚拟机。

状态: 已关机
配置文件: D:\ubuntu_16.04\ubuntu_16.04\Ubuntu 64 位.vmx
硬件兼容性: Workstation 14.x 虚拟机
主 IP 地址: 网络信息不可用

选择之前下载的 ubuntu 镜像文件，继续使用默认设置点击“下一步”。

图 2-6. ubuntu 虚拟机安装向导 6



安装客户机操作系统
虚拟机如同物理机，需要操作系统。您将如何安装客户机操作系统?

安装来源:

○ 安装程序光盘(D):
无可用驱动器

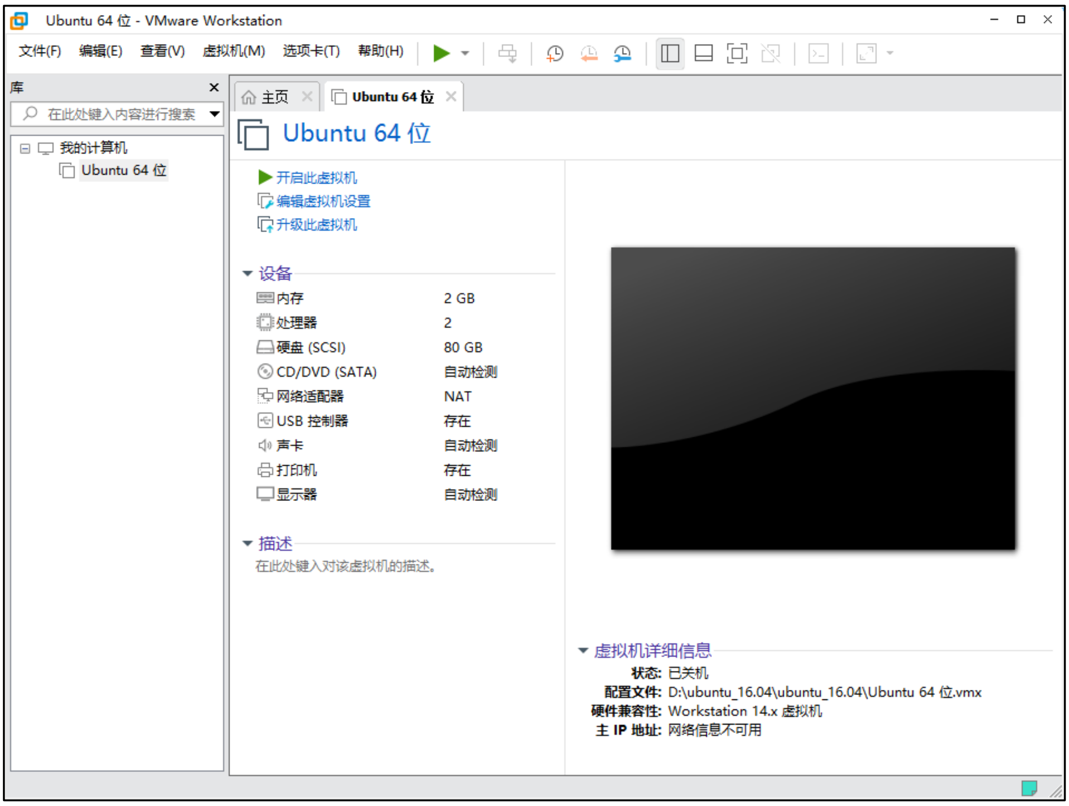
● 安装程序光盘映像文件(iso)(M):
浏览(R)...
⇨ 选择安装程序光盘映像继续。

○ 稍后安装操作系统(S)。
创建的虚拟机将包含一个空白硬盘。

虚拟机详细信息
状态: 已关机
配置文件: D:\ubuntu_16.04\ubuntu_16.04\Ubuntu 64 位.vmx
硬件兼容性: Workstation 14.x 虚拟机
主 IP 地址: 网络信息不可用

开始安装虚拟机系统，点击开始此虚拟机，第一次启动时间较长，选择默认设置直到启动完成即可。

图 2-7. ubuntu 虚拟机安装向导 7



Ubuntu 64 位

设备

内存	2 GB
处理器	2
硬盘 (SCSI)	80 GB
CD/DVD (SATA)	自动检测
网络适配器	NAT
USB 控制器	存在
声卡	自动检测
打印机	存在
显示器	自动检测

描述

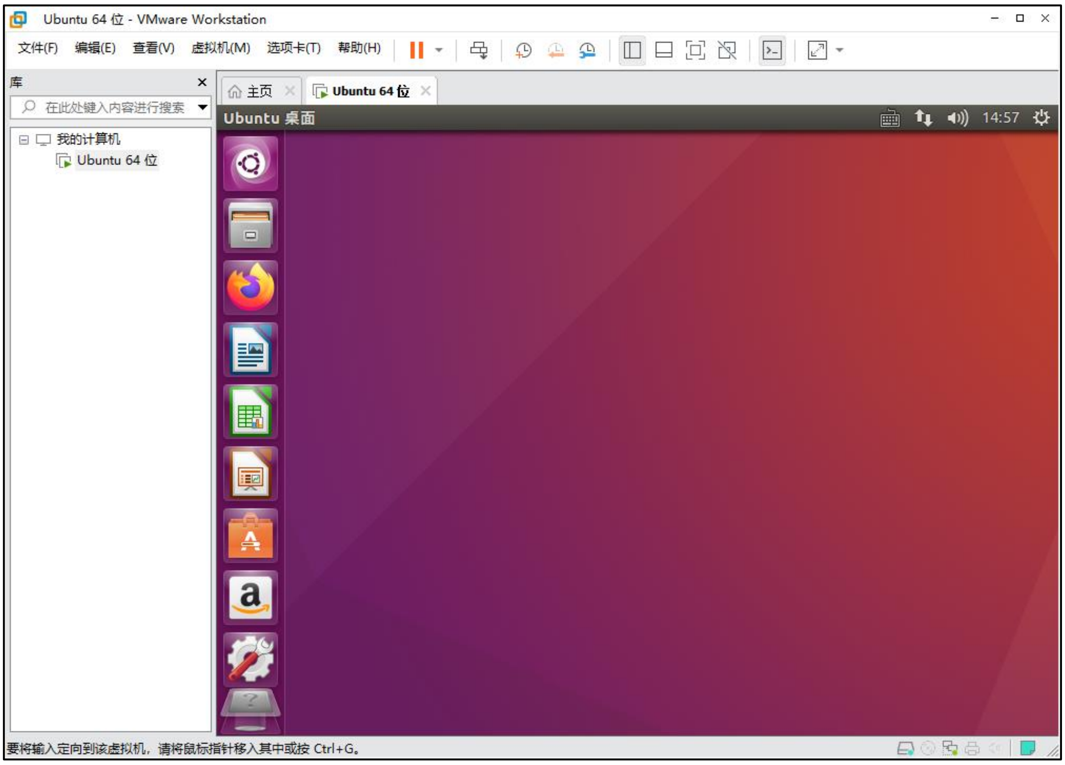
在此处键入对该虚拟机的描述。

虚拟机详细信息

状态: 已关机
配置文件: D:\ubuntu_16.04\ubuntu_16.04\Ubuntu 64 位.vmx
硬件兼容性: Workstation 14.x 虚拟机
主 IP 地址: 网络信息不可用

*图 2-8. ubuntu 虚拟机启动完成*是 ubuntu 虚拟机启动完成后的界面。

图 2-8. ubuntu 虚拟机启动完成


Ubuntu 64 位 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
库
在此处键入内容进行搜索
我的计算机
Ubuntu 64 位
主页
Ubuntu 64 位
Ubuntu 桌面
14:57

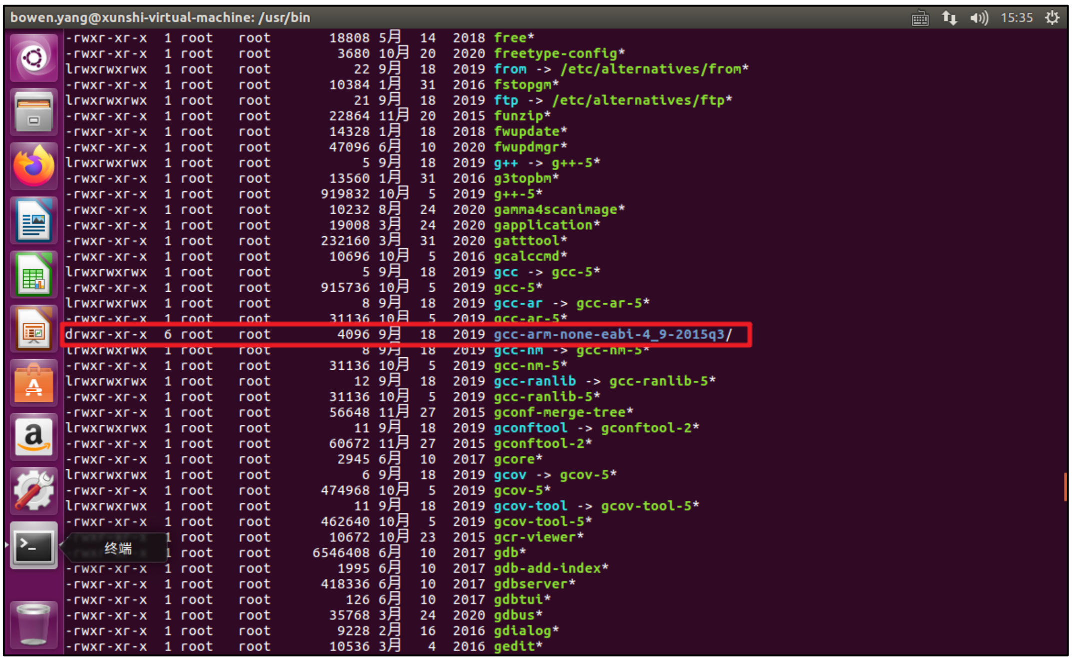
要将输入定向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。

2.2. 安装工具链

使用编译好的工具链，下载地址：https://launchpad.net/gcc-arm-embedded/+download

将下载好的工具链放到 ubuntu 的/usr/bin/目录下解压，解压后得到 gcc-arm-none-eabi-4_9-2015q3/目录。如*图 2-9. GCC 工具链所在目录*。

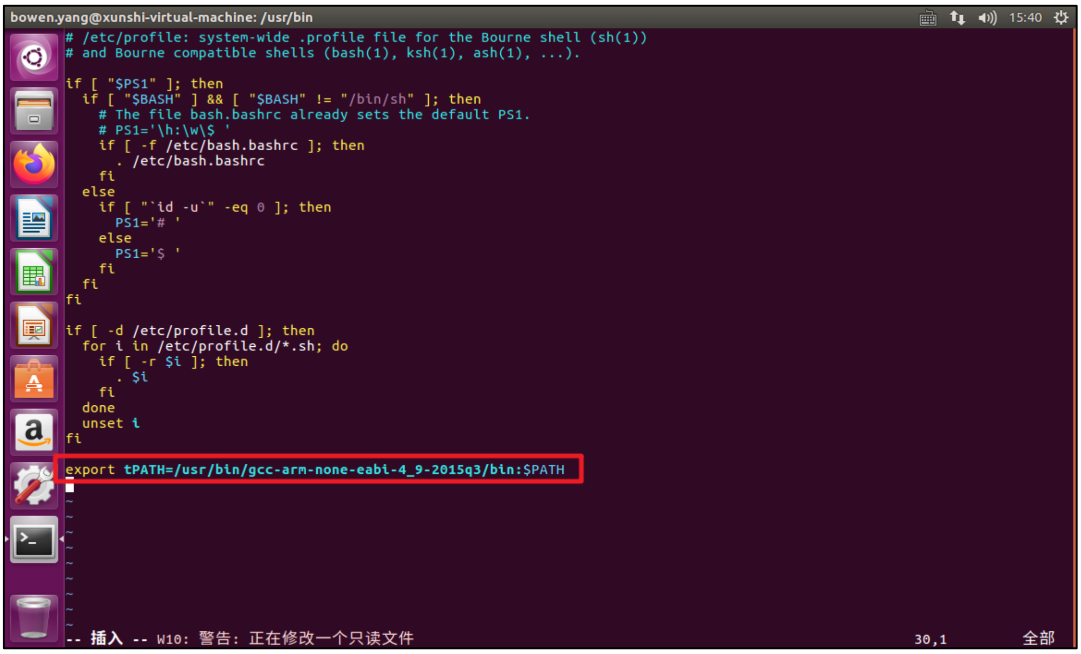
图 2-9. GCC 工具链所在目录



```
-rwxr-xr-x  1 root   root       18808 5月   14  2018 free*
-rwxr-xr-x  1 root   root        3680 10月  20  2020 freetype-config*
lrwxrwxrwx  1 root   root          22 9月   18  2019 from -> /etc/alternatives/from*
-rwxr-xr-x  1 root   root       10384 1月   31  2016 fstopgm*
lrwxrwxrwx  1 root   root          21 9月   18  2019 ftp -> /etc/alternatives/ftp*
-rwxr-xr-x  1 root   root       22864 11月  20  2015 funzip*
-rwxr-xr-x  1 root   root       14328 1月   18  2018 fwupdate*
-rwxr-xr-x  1 root   root       47096 6月   10  2020 fwupdmgr*
lrwxrwxrwx  1 root   root           5 9月   18  2019 g++ -> g++-5*
-rwxr-xr-x  1 root   root       13560 1月   31  2016 g3topbm*
-rwxr-xr-x  1 root   root      919832 10月   5  2019 g++-5*
-rwxr-xr-x  1 root   root       10232 8月   24  2020 gamma4scanimage*
-rwxr-xr-x  1 root   root       19008 3月   24  2020 gapplication*
-rwxr-xr-x  1 root   root      232160 3月   31  2020 gatttool*
-rwxr-xr-x  1 root   root       10696 10月   5  2016 gcalccmd*
lrwxrwxrwx  1 root   root           5 9月   18  2019 gcc -> gcc-5*
-rwxr-xr-x  1 root   root      915736 10月   5  2019 gcc-5*
lrwxrwxrwx  1 root   root           8 9月   18  2019 gcc-ar -> gcc-ar-5*
-rwxr-xr-x  1 root   root       31136 10月   5  2019 gcc-ar-5*
drwxr-xr-x  6 root   root        4096 9月   18  2019 gcc-arm-none-eabi-4_9-2015q3/
lrwxrwxrwx  1 root   root           8 9月   18  2019 gcc-nm -> gcc-nm-5*
-rwxr-xr-x  1 root   root       31136 10月   5  2019 gcc-nm-5*
lrwxrwxrwx  1 root   root          12 9月   18  2019 gcc-ranlib -> gcc-ranlib-5*
-rwxr-xr-x  1 root   root       31136 10月   5  2019 gcc-ranlib-5*
-rwxr-xr-x  1 root   root       56648 11月  27  2015 gconf-merge-tree*
lrwxrwxrwx  1 root   root          11 9月   18  2019 gconftool -> gconftool-2*
-rwxr-xr-x  1 root   root       60672 11月  27  2015 gconftool-2*
-rwxr-xr-x  1 root   root        2945 6月   10  2017 gcore*
lrwxrwxrwx  1 root   root           6 9月   18  2019 gcov -> gcov-5*
-rwxr-xr-x  1 root   root      474968 10月   5  2019 gcov-5*
lrwxrwxrwx  1 root   root          11 9月   18  2019 gcov-tool -> gcov-tool-5*
-rwxr-xr-x  1 root   root      462640 10月   5  2019 gcov-tool-5*
-rwxr-xr-x  1 root   root       10672 10月  23  2015 gcr-viewer*
-rwxr-xr-x  1 root   root     6546408 6月   10  2017 gdb*
-rwxr-xr-x  1 root   root        1995 6月   10  2017 gdb-add-index*
-rwxr-xr-x  1 root   root      418336 6月   10  2017 gdbserver*
-rwxr-xr-x  1 root   root         126 6月   10  2017 gdbtui*
-rwxr-xr-x  1 root   root       35768 3月   24  2020 gdbus*
-rwxr-xr-x  1 root   root        9228 2月   16  2016 gdialog*
-rwxr-xr-x  1 root   root       10536 3月    4  2016 gedit*
```

接下来配置 linux 系统环境变量，在/etc/profile 文件最后一行指定 GCC 工具链目录。

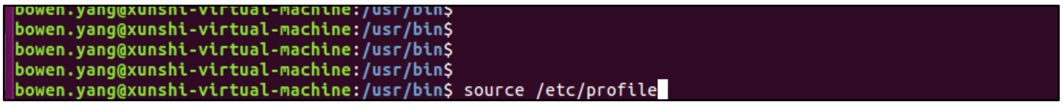
图 2-10. 配置环境变量



```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "$PS1" ]; then
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

export tPATH=/usr/bin/gcc-arm-none-eabi-4_9-2015q3/bin:$PATH

~
~
~
~
~
~
~
~
~
-- 插入 -- W10: 警告: 正在修改一个只读文件                        30,1          全部
```

添加完成后执行命令：source /etc/profile 使环境变量生效，不用重启虚拟机。

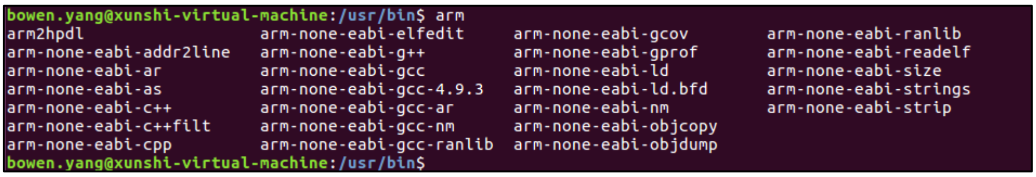
图 2-11. 环境变量生效



```
bowen.yang@xunshi-virtual-machine:/usr/bin$
bowen.yang@xunshi-virtual-machine:/usr/bin$
bowen.yang@xunshi-virtual-machine:/usr/bin$
bowen.yang@xunshi-virtual-machine:/usr/bin$
bowen.yang@xunshi-virtual-machine:/usr/bin$ source /etc/profile
```

完成后再任意目录下输入命令 arm 加 tab 键，如果系统自动弹出工具链则列表代表安装成功，如*图 2-12. 交叉工具链列表*，弹出的工具链列表中有我们需要的 arm-none-eabi-gcc 和 arm-none-eabi-objcop。

图 2-12. 交叉工具链列表



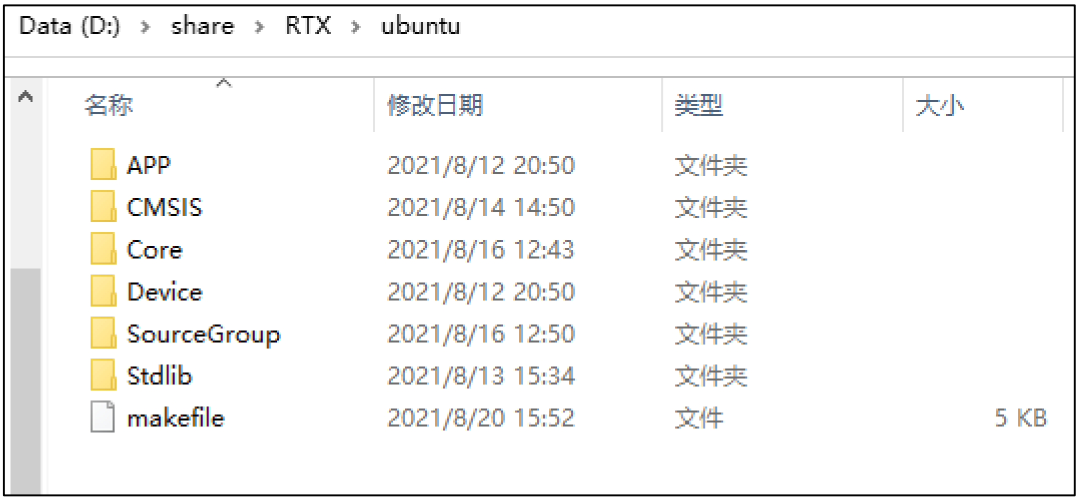
```
bowen.yang@xunshi-virtual-machine:/usr/bin$ arm
arm2hpdl                    arm-none-eabi-elfedit       arm-none-eabi-gcov          arm-none-eabi-ranlib
arm-none-eabi-addr2line     arm-none-eabi-g++           arm-none-eabi-gprof         arm-none-eabi-readelf
arm-none-eabi-ar            arm-none-eabi-gcc           arm-none-eabi-ld            arm-none-eabi-size
arm-none-eabi-as            arm-none-eabi-gcc-4.9.3     arm-none-eabi-ld.bfd        arm-none-eabi-strings
arm-none-eabi-c++           arm-none-eabi-gcc-ar        arm-none-eabi-nm            arm-none-eabi-strip
arm-none-eabi-c++filt       arm-none-eabi-gcc-nm        arm-none-eabi-objcopy
arm-none-eabi-cpp           arm-none-eabi-gcc-ranlib    arm-none-eabi-objdump
bowen.yang@xunshi-virtual-machine:/usr/bin$
```

3. 建立工程

3.1. 建立工程目录

将所需要编译的 RTOS 工程代码放到 D:\share\RTX\ubuntu，如图*图 3-1. 工程代码目录*。

图 3-1. 工程代码目录



Data (D:) › share › RTX › ubuntu

名称	修改日期	类型	大小
APP	2021/8/12 20:50	文件夹	
CMSIS	2021/8/14 14:50	文件夹	
Core	2021/8/16 12:43	文件夹	
Device	2021/8/12 20:50	文件夹	
SourceGroup	2021/8/16 12:50	文件夹	
Stdlib	2021/8/13 15:34	文件夹	
makefile	2021/8/20 15:52	文件	5 KB

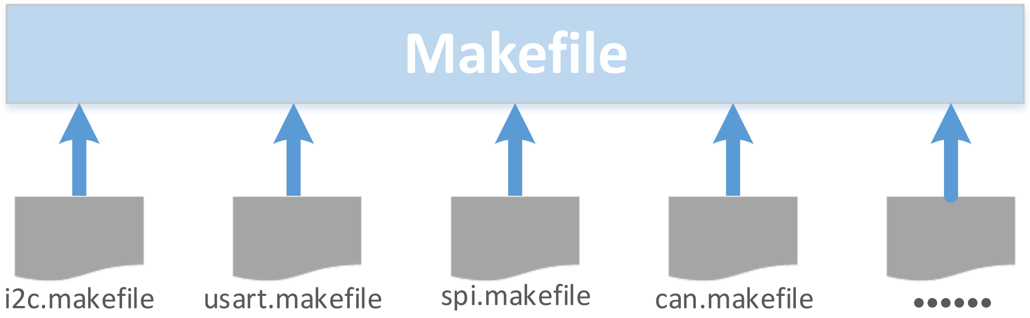
目录结构如下所示,每个 source 目录下都需要一个 makefile 文件。

```
├──APP
|   ├──include
|   └──source
├──CMSIS
|   ├──GCC
|   ├──include
|   └──source
├──Core
|   ├──include
|   └──source
├──Device
|   ├──include
|   └──source
└──SourceGroup
    ├──include
    └──source
```

3.2. Makefile 文件编写

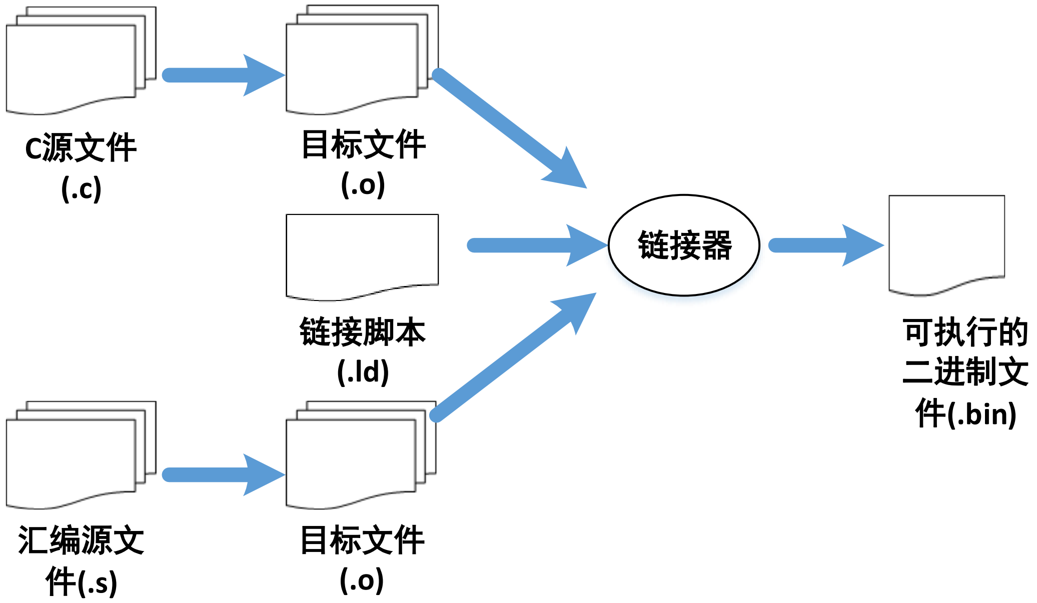
使用顶层 makefile 管理各子目录下的 makefile，如图*图 3-2. 图示*

图 3-2. 图示



整体编译流程如图*图 3-3. 编译流程*。

图 3-3. 编译流程



顶层 makefile 编写如下：

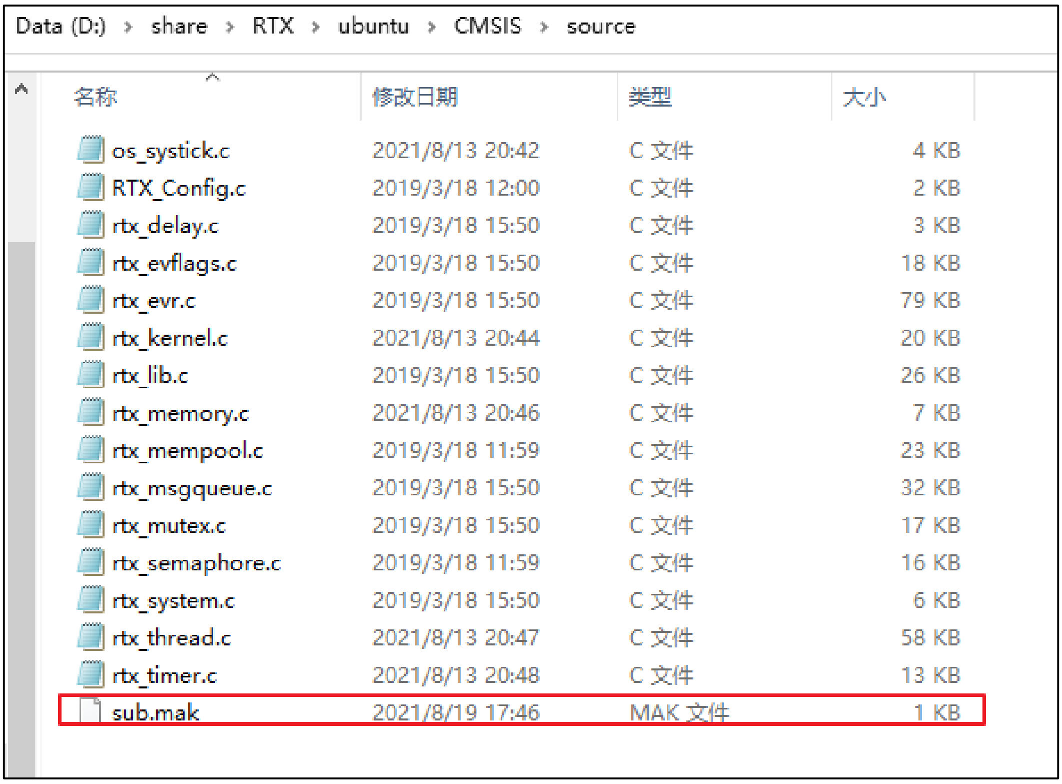
表 3-1. 顶层 makefile 编写

```
CROSS_COMPILE=arm-none-eabi-

CC              = $(CROSS_COMPILE)gcc
OBJCOPY         = $(CROSS_COMPILE)objcopy

TOP=$(shell pwd)
INC_FLAGS = -I$(TOP)/Device/include \
              -I$(TOP)/Core/include \
              -I$(TOP)/APP/include \
              -I$(TOP)/Stdlib/include \
              -I$(TOP)/CMSIS/include

CC_FLAGS   =   -W   -Wall   -g   -mcpu=cortex-m4   -mthumb   -D   GD32F30X_HD   -D
USE_STDPERIPH_DRIVER $(INC_FLAGS) -O0 -std=gnu11
```

```
CC_ASM_FLAGS = -mthumb -mcpu=cortex-m4 -g -Wa,--warn

CC_LD_FLAGS += -mthumb -mcpu=cortex-m4
CC_LD_FLAGS += -Wl,--start-group -lc -lm -Wl,--end-group -specs=nosys.specs -static -Wl,-cref,-
u,Reset_Handler-Wl,-Map=RTX_Project.map-Wl,--gc-sections-Wl,--
defsym=malloc_getpagesize_P=0x80

LD_PATH = -TDevice/source/gd32f30x_flash.ld
#用于指示该模块是否参与编译
SUPPORT_IIC = yes
SUPPORT_SPI = yes
SUPPORT_CAN = yes
SUPPORT_KEY = no
#指定需要编译的源文件
include APP/source/sub.mak
include CMSIS/source/sub.mak
include Device/source/sub.mak
include SourceGroup/source/sub.mak

TARGET = RTX_Project

.PHONY: clean all
#替换为.o
C_OBJ = $(C_SRC:%.c=%.o)
ASM_OBJ = $(ASM_SRC:%.s=%.o)

all:$(C_OBJ) $(ASM_OBJ)
	$(CC) $(C_OBJ) $(ASM_OBJ) $(LD_PATH) -o $(TARGET).elf $(CC_LD_FLAGS)
	$(OBJCOPY) $(TARGET).elf   $(TARGET).bin -Obinary

%.o:%.c
	$(CC) -c $(CC_FLAGS) -o $@ $<
%.o:%.s
	$(CC) -c $(CC_ASM_FLAGS) -o $@ $<

clean:
	rm -rf *.o $(C_OBJ) $(ASM_OBJ) $(TARGET) *.bin *.map *.elf
```

分别在 APP、CMSIS、Core、Device、SourceGroup 目录下建立 sub.mak 文件，所有目录下的 sub.mak 文件编写方式都一样，这里介绍 CMSIS/source 目录下的下的 sub.mak。

图 3-4. sub.mak 所在目录



Data (D:) › share › RTX › ubuntu › CMSIS › source

名称	修改日期	类型	大小
os_systick.c	2021/8/13 20:42	C 文件	4 KB
RTX_Config.c	2019/3/18 12:00	C 文件	2 KB
rtx_delay.c	2019/3/18 15:50	C 文件	3 KB
rtx_evflags.c	2019/3/18 15:50	C 文件	18 KB
rtx_evr.c	2019/3/18 15:50	C 文件	79 KB
rtx_kernel.c	2021/8/13 20:44	C 文件	20 KB
rtx_lib.c	2019/3/18 15:50	C 文件	26 KB
rtx_memory.c	2021/8/13 20:46	C 文件	7 KB
rtx_mempool.c	2019/3/18 11:59	C 文件	23 KB
rtx_msgqueue.c	2019/3/18 15:50	C 文件	32 KB
rtx_mutex.c	2019/3/18 15:50	C 文件	17 KB
rtx_semaphore.c	2019/3/18 11:59	C 文件	16 KB
rtx_system.c	2019/3/18 15:50	C 文件	6 KB
rtx_thread.c	2021/8/13 20:47	C 文件	58 KB
rtx_timer.c	2021/8/13 20:48	C 文件	13 KB
sub.mak	2021/8/19 17:46	MAK 文件	1 KB

在 sub.mak 文件中指定本目录中需要编译的文件，makefile 编写如下:

表 3-2. 子目录 sub.mak 编写

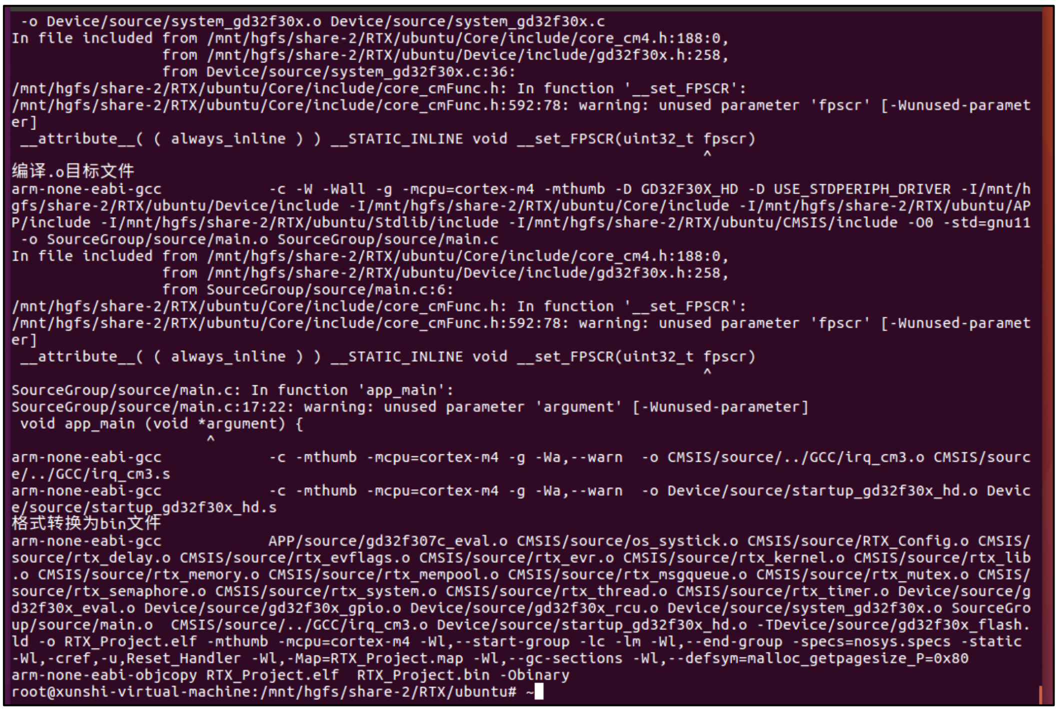
```
CMSIS_PATH = CMSIS/source
C_SRC += $(CMSIS_PATH)/os_systick.c \
              $(CMSIS_PATH)/RTX_Config.c \
              $(CMSIS_PATH)/rtx_delay.c \
              $(CMSIS_PATH)/rtx_evflags.c\
              $(CMSIS_PATH)/rtx_evr.c \
              $(CMSIS_PATH)/rtx_kernel.c \
              $(CMSIS_PATH)/rtx_lib.c \
              $(CMSIS_PATH)/rtx_memory.c \
              $(CMSIS_PATH)/rtx_mempool.c \
              $(CMSIS_PATH)/rtx_msgqueue.c \
              $(CMSIS_PATH)/rtx_mutex.c \
              $(CMSIS_PATH)/rtx_semaphore.c \
              $(CMSIS_PATH)/rtx_system.c \
              $(CMSIS_PATH)/rtx_thread.c \
              $(CMSIS_PATH)/rtx_timer.c

ifeq ($(SUPPORT_KEY), yes)
C_SRC += $(AUDIO_PATH)/rtx_keymanager.c
endif
```

```
ASM_SRC += $(CMSIS_PATH)/../GCC/irq_cm3.s
```

3.3. 编译和测试

在顶层 makefile 所在目录执行命令：make

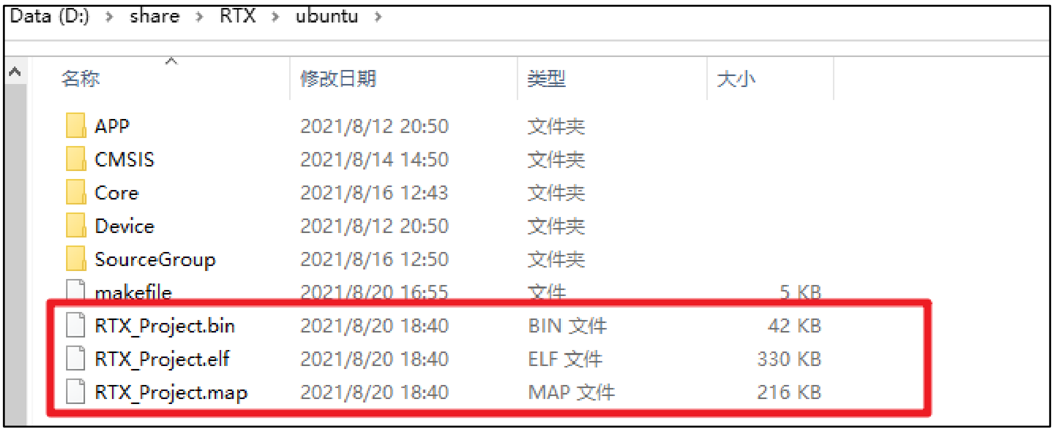
图 3-5. Mak 执行结果



```
 -o Device/source/system_gd32f30x.o Device/source/system_gd32f30x.c
In file included from /mnt/hgfs/share-2/RTX/ubuntu/Core/include/core_cm4.h:188:0,
                 from /mnt/hgfs/share-2/RTX/ubuntu/Device/include/gd32f30x.h:258,
                 from Device/source/system_gd32f30x.c:36:
/mnt/hgfs/share-2/RTX/ubuntu/Core/include/core_cmFunc.h: In function '__set_FPSCR':
/mnt/hgfs/share-2/RTX/ubuntu/Core/include/core_cmFunc.h:592:78: warning: unused parameter 'fpscr' [-Wunused-paramet
er]
 __attribute__( ( always_inline ) ) __STATIC_INLINE void __set_FPSCR(uint32_t fpscr)
                                                                              ^
编译.o目标文件
arm-none-eabi-gcc            -c -W -Wall -g -mcpu=cortex-m4 -mthumb -D GD32F30X_HD -D USE_STDPERIPH_DRIVER -I/mnt/h
gfs/share-2/RTX/ubuntu/Device/include -I/mnt/hgfs/share-2/RTX/ubuntu/Core/include -I/mnt/hgfs/share-2/RTX/ubuntu/AP
P/include -I/mnt/hgfs/share-2/RTX/ubuntu/Stdlib/include -I/mnt/hgfs/share-2/RTX/ubuntu/CMSIS/include -O0 -std=gnu11
 -o SourceGroup/source/main.o SourceGroup/source/main.c
In file included from /mnt/hgfs/share-2/RTX/ubuntu/Core/include/core_cm4.h:188:0,
                 from /mnt/hgfs/share-2/RTX/ubuntu/Device/include/gd32f30x.h:258,
                 from SourceGroup/source/main.c:6:
/mnt/hgfs/share-2/RTX/ubuntu/Core/include/core_cmFunc.h: In function '__set_FPSCR':
/mnt/hgfs/share-2/RTX/ubuntu/Core/include/core_cmFunc.h:592:78: warning: unused parameter 'fpscr' [-Wunused-paramet
er]
 __attribute__( ( always_inline ) ) __STATIC_INLINE void __set_FPSCR(uint32_t fpscr)
                                                                              ^

SourceGroup/source/main.c: In function 'app_main':
SourceGroup/source/main.c:17:22: warning: unused parameter 'argument' [-Wunused-parameter]
 void app_main (void *argument) {
                      ^

arm-none-eabi-gcc            -c -mthumb -mcpu=cortex-m4 -g -Wa,--warn  -o CMSIS/source/../GCC/irq_cm3.o CMSIS/sourc
e/../GCC/irq_cm3.s
arm-none-eabi-gcc            -c -mthumb -mcpu=cortex-m4 -g -Wa,--warn  -o Device/source/startup_gd32f30x_hd.o Devic
e/source/startup_gd32f30x_hd.s
格式转换为bin文件
arm-none-eabi-gcc            APP/source/gd32f307c_eval.o CMSIS/source/os_systick.o CMSIS/source/RTX_Config.o CMSIS/
source/rtx_delay.o CMSIS/source/rtx_evflags.o CMSIS/source/rtx_evr.o CMSIS/source/rtx_kernel.o CMSIS/source/rtx_lib
.o CMSIS/source/rtx_memory.o CMSIS/source/rtx_mempool.o CMSIS/source/rtx_msgqueue.o CMSIS/source/rtx_mutex.o CMSIS/
source/rtx_semaphore.o CMSIS/source/rtx_system.o CMSIS/source/rtx_thread.o CMSIS/source/rtx_timer.o Device/source/g
d32f30x_eval.o Device/source/gd32f30x_gpio.o Device/source/gd32f30x_rcu.o Device/source/system_gd32f30x.o SourceGro
up/source/main.o  CMSIS/source/../GCC/irq_cm3.o Device/source/startup_gd32f30x_hd.o -TDevice/source/gd32f30x_flash.
ld -o RTX_Project.elf -mthumb -mcpu=cortex-m4 -Wl,--start-group -lc -lm -Wl,--end-group -specs=nosys.specs -static
-Wl,-cref,-u,Reset_Handler -Wl,-Map=RTX_Project.map -Wl,--gc-sections -Wl,--defsym=malloc_getpagesize_P=0x80
arm-none-eabi-objcopy RTX_Project.elf  RTX_Project.bin -Obinary
root@xunshi-virtual-machine:/mnt/hgfs/share-2/RTX/ubuntu# ~
```

编译成功之后在 makefile 同级目录下会产生编译出的.bin 文件，.elf 文件和.map 文件。

图 3-6. 顶层 makefile 所在目录



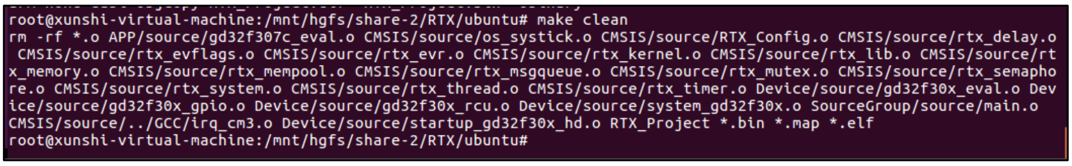
Data (D:) › share › RTX › ubuntu ›

名称	修改日期	类型	大小
APP	2021/8/12 20:50	文件夹	
CMSIS	2021/8/14 14:50	文件夹	
Core	2021/8/16 12:43	文件夹	
Device	2021/8/12 20:50	文件夹	
SourceGroup	2021/8/16 12:50	文件夹	
makefile	2021/8/20 16:55	文件	5 KB
RTX_Project.bin	2021/8/20 18:40	BIN 文件	42 KB
RTX_Project.elf	2021/8/20 18:40	ELF 文件	330 KB
RTX_Project.map	2021/8/20 18:40	MAP 文件	216 KB

在 makefile 目录下执行命令：make clean

会发现之前执行 make 命令生成的.bin 文件，.elf 文件和.map 文件都已被删除。

图 3-7. Make clean 执行结果



```
root@xunshi-virtual-machine:/mnt/hgfs/share-2/RTX/ubuntu# make clean
rm -rf *.o APP/source/gd32f307c_eval.o CMSIS/source/os_systick.o CMSIS/source/RTX_Config.o CMSIS/source/rtx_delay.o
 CMSIS/source/rtx_evflags.o CMSIS/source/rtx_evr.o CMSIS/source/rtx_kernel.o CMSIS/source/rtx_lib.o CMSIS/source/rt
x_memory.o CMSIS/source/rtx_mempool.o CMSIS/source/rtx_msgqueue.o CMSIS/source/rtx_mutex.o CMSIS/source/rtx_semapho
re.o CMSIS/source/rtx_system.o CMSIS/source/rtx_thread.o CMSIS/source/rtx_timer.o Device/source/gd32f30x_eval.o Dev
ice/source/gd32f30x_gpio.o Device/source/gd32f30x_rcu.o Device/source/system_gd32f30x.o SourceGroup/source/main.o
CMSIS/source/../GCC/irq_cm3.o Device/source/startup_gd32f30x_hd.o RTX_Project *.bin *.map *.elf
root@xunshi-virtual-machine:/mnt/hgfs/share-2/RTX/ubuntu#
```

最后为了验证编译出来的固件是否可以正常运行，使用 SEGGR J-Flash 将编译出的.bin 文件烧写进 MCU 进行测试。观察 LED 灯可正常运行。

4. 版本历史

表 4-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2021 年 8 月 26 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.

© 2021 GigaDevice – All rights reserved