


# UEFI Development Exploration 103-UEFI Self-Certification Tests (SCT)

 luobing4365    Modified on 2024-12-07 11:39:19     Read 766     Collection 17     Likes 14    copyright

Category Column: UEFI Development    Article Tags: UEFI SCT UEFI programming and practice

 **UEFI Development**    This column includes this content    503 Subscribe    104 articles    Subscribe to

our column

Please keep it-> Author: Luo Bing <https://blog.csdn.net/luobing4365> )

## UEFI Development Exploration 103-UEFI Self-Certification Tests

- 1 Overview
- 2 Compilation
- 3 Use
- 4 More

### Overview

Some netizens asked about the compilation of SCT in the WeChat group. I am usually too busy at work and don't have time to read it. I have some time to myself on weekends when I am with my kids, so I just read it.

I have read about it before, but I have not studied and experimented with it in detail. When I was dealing with the Arm System Ready tool ACS, I felt that the two were somewhat similar, and I wondered if they originated from this project.

Reference documents and [git](#) repositories:

- UEFI SCT · tianocore/tianocore.github.io Wiki · GitHub
- Test Tools | Unified Extensible Firmware Interface Forum
- Self Certification Test (SCT) II — Self Certification Test (SCT) II
- GitHub - tianocore/edk2-test: Test infrastructure and test cases for EDK II based firmware

It is recommended to refer to the UEFI official website (the second link above). The documents on github.io are a bit old.

### Compilation

Just check how to compile. (Downloading code directly from GitHub is very slow, so I still use the old method, syncing to gitee and then pulling it down to compile. For the specific method, you can see the introduction in my

previous blog)

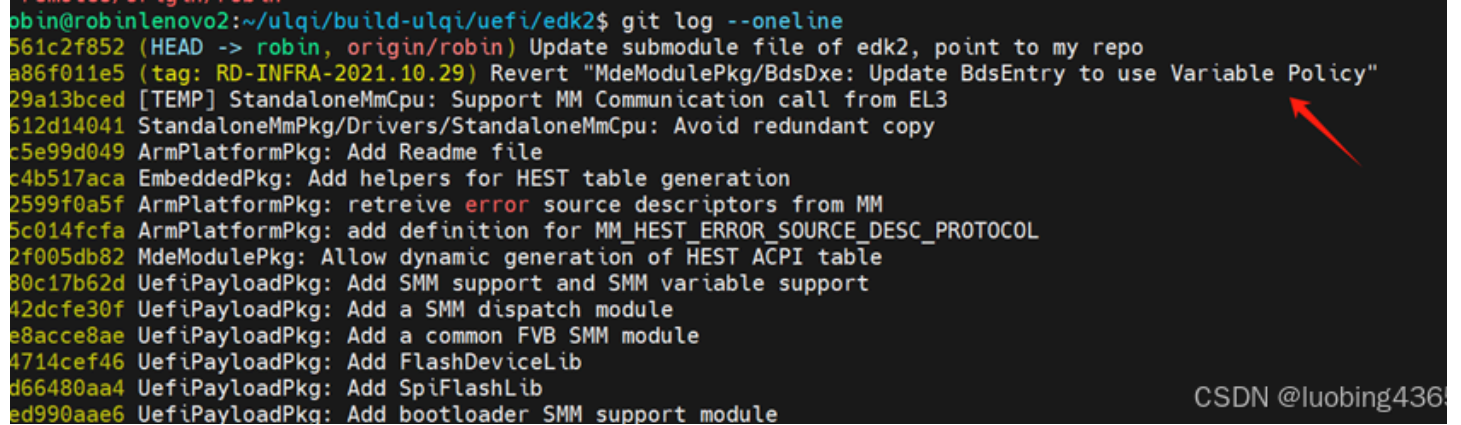
A thoughtful engineer gave me the compilation steps for AARCH64 (thanks to Rebecca Cran for his work, document location: edk2-test/uefi-sct/HowToBuild/HowToBuildSct.txt). The compilation for X86 is probably similar but I didn't try it:

```

1 Build Instructions for UEFI SCTII AARCH64 (Linux)
2
3 Pre-requisite: install required tools. For example, on WSL running Ubuntu:
4 $ sudo apt-get install build-essential uuid-dev python3-distutils gcc-aarch64-linux-gnu
5
6 1) mkdir "sct_workspace"
7 2) cd sct_workspace
8 3) git clone https://github.com/tianocore/edk2-test.git
9 4) git clone --recursive https://github.com/tianocore/edk2.git
10 5) ln -s edk2-test/uefi-sct/SctPkg/ SctPkg
11 6) SctPkg/build.sh AARCH64 GCC RELEASE

```

I checked the UEFI compilation framework I often use, and the tag used by edk2 is:



```

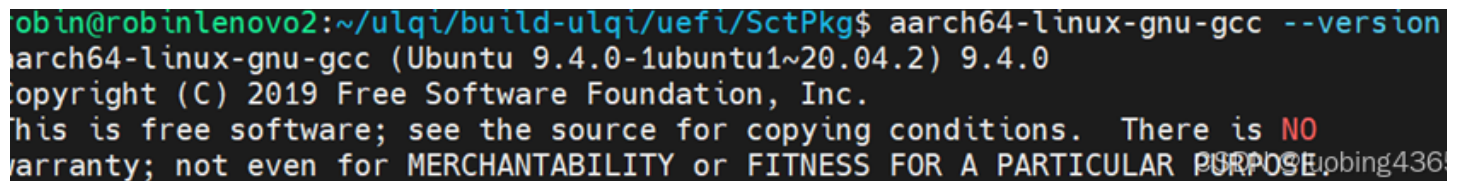
robin@robinlenovo2:~/ulqi/build-ulqi/uefi/edk2$ git log --oneline
561c2f852 (HEAD -> robin, origin/robin) Update submodule file of edk2, point to my repo
a86f011e5 (tag: RD-INFRA-2021.10.29) Revert "MdeModulePkg/BdsDxe: Update BdsEntry to use Variable Policy"
29a13bced [TEMP] StandaloneMmCpu: Support MM Communication call from EL3
612d14041 StandaloneMmPkg/Drivers/StandaloneMmCpu: Avoid redundant copy
c5e99d049 ArmPlatformPkg: Add Readme file
c4b517aca EmbeddedPkg: Add helpers for HEST table generation
2599f0a5f ArmPlatformPkg: retrieve error source descriptors from MM
5c014fcfa ArmPlatformPkg: add definition for MM HEST ERROR SOURCE_DESC_PROTOCOL
2f005db82 MdeModulePkg: Allow dynamic generation of HEST ACPI table
80c17b62d UefiPayloadPkg: Add SMM support and SMM variable support
42dcfe30f UefiPayloadPkg: Add a SMM dispatch module
e8acce8ae UefiPayloadPkg: Add a common FVB SMM module
4714cef46 UefiPayloadPkg: Add FlashDeviceLib
d66480aa4 UefiPayloadPkg: Add SpiFlashLib
ed990aae6 UefiPayloadPkg: Add bootloader SMM support module

```

CSDN @luobing436

I use edk2 on the official Arm library, which is slightly different from the main line on tianocore. However, the difference should not be big, and I did not check the difference carefully.

As for the compiler, I directly use gcc-aarch64-linux-gnu, which I installed when I read Uncle Ben's linux book



```

robin@robinlenovo2:~/ulqi/build-ulqi/uefi/SctPkg$ aarch64-linux-gnu-gcc --version
aarch64-linux-gnu-gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

```

CSDN @luobing436

Follow the above steps and compile directly:

```

bin@robinlenovo2:~/ulqi/build-ulqi/uefi$ SctPkg/build.sh AARCH64 GCC DEBUG
rget: AARCH64
ld: other
OLCHAIN is GCC5
olchain prefix: GCC5_AARCH64_PREFIX=aarch64-linux-gnu-
initializing workspace
adding previous configuration from /home/robin/ulqi/build-ulqi/uefi/edk2/Conf/BuildEnv.sh
ing EDK2 in-source Basetools
RKSPACE: /home/robin/ulqi/build-ulqi/uefi
K_TOOLS_PATH: /home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools
NF_PATH: /home/robin/ulqi/build-ulqi/uefi/edk2/Conf
ing prebuilt tools
lding GenBin
ke: Entering directory '/home/robin/ulqi/build-ulqi/uefi/edk2-test/uefi-sct/SctPkg/Tools/Source/GenBin'
tempting to detect ARCH from 'uname -m': x86_64
ected ARCH of X64 using uname.
c -c -I /home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools/Source/C -I /home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools/Source/C/Include/Common -I /home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools/Source/C/Include/IndustryStandard -I /home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools/Source/C/Common/ -I ... -I ... -I /home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools/Source/C/Include/X64/ -MD -fshort-wchar -fno-strict-aliasing -fwrapv -fno-delete-null-pointer-checks -Wall -Werror -Wno-deprecated-declarations -Wno-strict-aliasing -Wno-unused-result -nostdlib -g -O2 GenBin.c -o GenBin.o
c -o /home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools/Source/C/bin/GenBin GenBin.o -L/home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools/Source/C/libs
ke: Leaving directory '/home/robin/ulqi/build-ulqi/uefi/edk2-test/uefi-sct/SctPkg/Tools/Source/GenBin'
ld environment: Linux-4.4.0-22621-Microsoft-x86_64-with-glibc2.29
ld start time: 09:55:30, Dec.07 2024

RKSPACE      = /home/robin/ulqi/build-ulqi/uefi
CKAGES_PATH  = /home/robin/ulqi/build-ulqi/uefi/edk2:/home/robin/ulqi/build-ulqi/uefi/SctPkg
K_TOOLS_PATH  = /home/robin/ulqi/build-ulqi/uefi/edk2/BaseTools
NF_PATH       = /home/robin/ulqi/build-ulqi/uefi/edk2/Conf
THON_COMMAND  = /usr/bin/python3.8

rocessing meta-data .
chitecture(s) = AARCH64
ild target    = DEBUG
olchain       = GCC5

tive Platform = /home/robin/ulqi/build-ulqi/uefi/SctPkg/UEFI/UEFI_SCT.dsc
..... done!
lding ... /home/robin/ulqi/build-ulqi/uefi/edk2/ArmPkg/Library/CompilerIntrinsicsLib/CompilerIntrinsicsLib.inf [AARCH64]
lding ... /home/robin/ulqi/build-ulqi/uefi/SctPkg/Library/EfiTestLib/EfiTestLib.inf [AARCH64]
lding ... /home/robin/ulqi/build-ulqi/uefi/edk2/MdePkg/Library/UefiDriverEntryPoint/UefiDriverEntryPoint.inf [AARCH64]

```

CSDN @luobing436.

he script seems to only provide RELEASE and DEBUG, and does not provide NOOPT support. If you need debugging, you can add support in the script, and DEBUG will still optimize some code.

## Use

CT is just an application and is relatively simple to use. For specific parameters and usage methods, please refer to the corresponding document (edk2-test/uefi-sct/Doc/UserGuide).

directly use the framework I usually use, start UEFI+Qemu, and run it directly:

```

S0:\> driver at 0x00046E2B000 EntryPoint=0x00046E519D0 SCT.efi
installProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 40844D98
ProtectUefiImageCommon - 0x40844240
- 0x00000000046E2B000 - 0x0000000000003C000
SetUefiImageMemoryAttributes - 0x00000000046E2B000 - 0x0000000000001000 (0x0000000000004008)
SetUefiImageMemoryAttributes - 0x00000000046E2C000 - 0x00000000000034000 (0x00000000000020008)
SetUefiImageMemoryAttributes - 0x00000000046E60000 - 0x0000000000007000 (0x0000000000004008)
installProtocolInterface: 752F3136-4E16-4FDC-A22A-E5F46812F4CA 46FD44B0
S0open: Created new directory entry 'Sct.log'
S0open: Open '\Sct.log' Success
S0open: Created new directory entry 'Sct.log'
S0open: Open '\Sct.log' Success
EFI2.7 Self Certification Test(SCT2)

```

sage:

```
CT [-a | -c | -s <seq> | -u | -p <MNP | IP4 | SERIAL>] [-r] [-g <report>][-v]
```

- a Executes all test cases.
- c Continues execute the test cases.
- g Generates test report.
- p Passive Mode with specified communication layer
- r Resets all test results.
- s Executes the test cases in the test sequence file.
- u Turns into user-friendly interface.
- f Force the operation execution, no confirmation from user.
- v Verbose function to disable screen output.

one!

CSDN @luobing436.

he UEFI (VirtQemu) I use is compiled with NOOPT, so there will be a lot of debugging information printed out, on't worry about it. It is obvious that SCT can already run.

More details can be explored on this basis.

Attached is the Qemu script:

```
1 | qemu-system-aarch64 \
2 | -M virt,gic-version=3 \
3 | -cpu cortex-a72 \
4 | -bios QEMU_EFI.fd \
5 | -hda fat:rw:.../..../uefi_sh_app/ \
6 | -net none \
7 | -serial mon:telnet:::8888,server,nowait -nographic
```

## More

Recently, I have been asked many questions by netizens, many of which are about basic issues such as compilation and debugging.

These things are not particularly technical, and are background knowledge that is easy for those who know how to do them and difficult for those who don't. In view of this, I have set up a small plan. Based on the framework I am currently using, I want to transform an overall environment for UEFI and Linux debugging. The general goals are as follows:

1) Use Qemu to build hardware simulation;

2) Support UEFI (Arm architecture uses VirtQemu, X86 uses Ovmf, etc.), Linux, qemu download and compile, and use script control;

3) Support automatic packaging of Linux into a boot image.

Based on this framework, you can do a lot of UEFI and Linux experiments. The specific completion time is not set, but it will be done around March. There is plenty of time to do this during the Chinese New Year.