

UEFI Development Exploration 97 – EDK2 Simulator to Build a Network Environment

原创 luobing4365 ⌚ Posted on 2021-08-22 09:36:09 👁 Read 2.6k ⭐ Collection 10 👍 Likes 3

Copyright CC 4.0 BY-SA

Category columns: UEFI Development Article Tags: uefi bios UEFI Programming Practice Low-level application development EDK2



UEFI Development This column includes this content

104 articles

Subscribe to

our column

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

EDK2 simulator builds network environment

1. Build EDK2 development environment

- 1) Tool Installation
- 2) Download the code library
- 3) Update submodules
- 4) Compilation tools

2. Build a network test environment

- 1) Install Winpcap
- 2) Compile SnpNt32Io
- 3) Compile EDK2 simulator
- 4) Configure the simulator network environment

3 Network Program Testing

Recently, many netizens have asked questions related to the Internet. Most of the issues discussed are about setting up the network environment, and some of the phenomena I have never encountered.

In UEFI Development Exploration 49, I introduced building a UEFI network environment in the Nt32 simulator and a real environment. The EDK2 I used at that time was the UDK2018 version.

With the upgrade of EDK2, the original NT32 simulator (compiled with Nt32Pkg) has been canceled and replaced by EmulatorPkg. The latter provides more comprehensive functions, including 32-bit and 64-bit simulators, which have been introduced in previous blogs.

Therefore, this article is going to use the simulator compiled by EmulatorPkg to rebuild the network test environment. The previous simulator was compiled with Nt32Pkg, so I named it Nt32 simulator; the current EmulatorPkg name is too long, so for the convenience of writing, I call it EDK2 simulator.

1. Build EDK2 development environment

For detailed construction process, please refer to UEFI Development Exploration 22 and 57. Here is a brief description of the construction process. The steps are as follows:

1) Tool Installation

Install necessary development tools, including Visual Studio, Python, ASL, and Nasm;

2) Download the code library

Use Git to download the EDK2 code and StdLib code to your local computer.

Create a new directory edk202011 on the C drive and use the following command to download:

c	AI generated projects	登录复制	run
1	robin@robin-PC MINGW64 /c/edk202011		
2	\$ git clone --branch stable/202011 https://gitee.com/luobing4365/edk2.git		
3	\$ git clone https://gitee.com/luobing4365/edk2-libc.git		

Please note that the download address here is my private repository. I synchronized the edk2 and edk2-libc repositories on github to my repository on gitee for your own download.

Don't download it according to the command I gave. The original repositories are <https://github.com/tianocore/edk2.git> and <https://github.com/tianocore/edk2-libc.git>. You can use the following command to download:

c	AI generated projects	登录复制	run
1	\$ git clone --branch stable/202011 https://github.com/tianocore/edk2.git		
2	\$ git clone https://github.com/tianocore/edk2-libc.git		

Of course, depending on your internet speed, the download speed will be slow. It is recommended to use the same method to synchronize the repository on GitHub to gitee, or directly find the same repository on gitee and download it.

The branch used in this article is stable/202011, and subsequent experimental results are all based on this version.

3) Update submodules

Modify .gitmodules in edk2 and change the submodule.

If you can use GitHub to download directly, you don't need to modify .gitmodules, just run the subsequent update commands directly. The content I modified is as follows:

c	AI generated projects	登录复制	run
1	[submodule "CryptoPkg/Library/OpensslLib/openssl"]		
2	path = CryptoPkg/Library/OpensslLib/openssl		
3	url = https://gitee.com/luobing4365/openssl.git		
4	[submodule "SoftFloat"]		
5			

```

6   path = ArmPkg/Library/ArmSoftFloatLib/berkeley-softfloat-3
7   url = https://gitee.com/luobing4365/berkeley-softfloat-3.git
8 [submodule "UnitTestFrameworkPkg/Library/CmockaLib/cmocka"]
9   path = UnitTestFrameworkPkg/Library/CmockaLib/cmocka
10  url = https://gitee.com/luobing4365/cmocka.git
11 [submodule "MdeModulePkg/Universal/RegularExpressionDxe/oniguruma"]
12  path = MdeModulePkg/Universal/RegularExpressionDxe/oniguruma
13  url = https://gitee.com/luobing4365/oniguruma.git
14 [submodule "MdeModulePkg/Library/BrotliCustomDecompressLib/brotli"]
15  path = MdeModulePkg/Library/BrotliCustomDecompressLib/brotli
16  url = https://gitee.com/luobing4365/brotli.git
17 [submodule "BaseTools/Source/C/BrotliCompress/brotli"]
18  path = BaseTools/Source/C/BrotliCompress/brotli
19  url = https://gitee.com/luobing4365/brotli.git
   ignore = untracked

```

收起 ^

Once again, the addresses given for the submodules above are the addresses where I import the submodule repositories on GitHub to gitee. You can refer to my method, but don't use them directly, as these are my private addresses and cannot be used directly.

For specific import methods, please refer to the content of UEFI Development Exploration 57.

Then use the following command to update the submodule:

c

AI generated projects

登录复制

run

```
1 | $ git submodule update --init
```

4) Compilation tools

Copy all three folders in `ekd2-libc`: `AppPkg`, `StdLib` and `StdLibPrivateInternalFiles` to the `edk2` directory to facilitate subsequent compilation. (Of course, you don't have to do this. The method has been introduced in the previous blog, please check it out yourself)

Open the Visual Studio Native command line (such as "VS2015 x86 Native Tools Command Prompt") and use the following command to compile `BaseTools` and other tools:

c

AI generated projects

登录复制

run

```
1 | C:\edk202011\edk2> edksetup.bat Rebuild
```

After compilation is complete, `BaseTools` and other tools will be automatically generated.

You can use the following command to test whether the development environment is successfully built:

c

AI generated projects

登录复制

run

```
1 | C:\edk202011\edk2> build -p AppPkg\AppPkg.dsc -a IA32
```

You will find an error message. This is mainly because there is a sentence on line 31 of AppPkg (I am using the latest StdLib library - 20210821):

c

AI generated projects

登录复制

run

```
1 | !include MdePkg/MdeLibs.dsc.inc
```

This file is not found in the source code of edk2. Add a "#" symbol before the statement to comment it out and compile AppPkg again. You can see that the compilation is successful, indicating that the development environment has been set up.

2. Build a network test environment

The network test environment built this time is only for the 32-bit simulation environment of EDK2. This is mainly because the NetNt32Io file provided by Intel only supports 32-bit. Although the EDK2 simulation environment (EmulatorPkg compilation) supports 32-bit and 64-bit, it is better to test 64-bit network programs on the actual machine.

The construction steps are as follows:

1) Install Winpcap

Winpcap is a professional software for network packet capture. It is a free and public network access system. It can provide win32 applications with the ability to access the underlying network. In the simulator, it is equivalent to the driver of the network card. Download address:

<https://www.winpcap.org/default.htm>.

This tool must be installed. The network packets sent by the simulator will be captured and forwarded by Winpcap.

2) Compile SnpNt32Io

Download the source code from github to your local computer: <https://github.com/tianocore/edk2-NetNt32Io>.

Create a folder NetNt32Io in drive C and copy the source code into it.

Then download the Winpcap development package WpdPack, the download address is: <https://www.winpcap.org/devel.htm>. Unzip WpdPack and copy it to the folder NetNt32Io just created. The created directory is as follows:

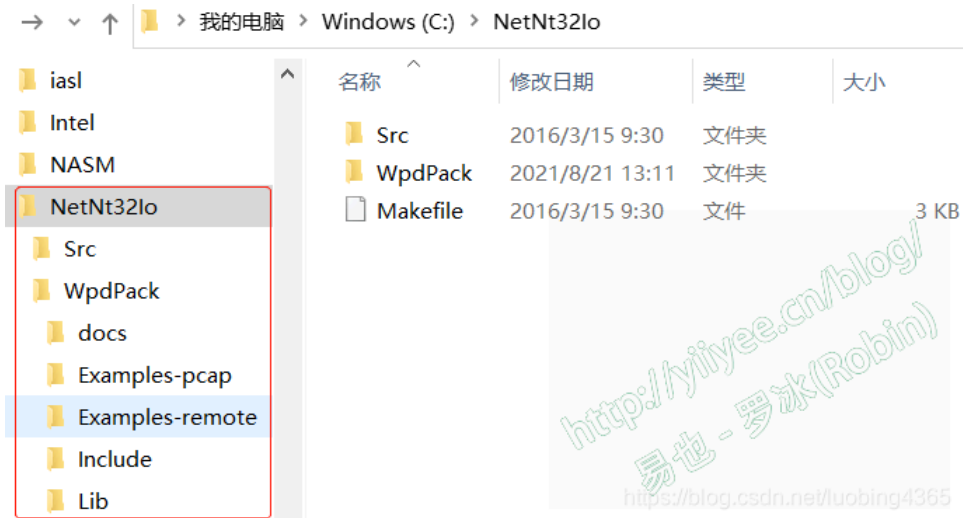


Figure 1 SnpNt32Io compilation directory structure

Open the Visual Studio command line (the same command line as used to compile the UEFI code), enter the source code directory, and enter the following command:

```
c                                                                    AI generated projects  登录复制  run
1 | C:\NetNt32Io> nmake TARGET=RELEASE
```

The directory Release_IA32 will be automatically generated in the NetNt32Io folder, which contains the SnpNt32Io.dll we need.

3) Compile EDK2 simulator

Compile the 32-bit simulator as follows:

```
c                                                                    AI generated projects  登录复制  run
1 | C:\edk202011\edk2> edksetup.bat
2 | C:\edk202011\edk2> build -p EmulatorPkg\ EmulatorPkg.dsc -a IA32
```

The compiled simulator is located in the folder C:\edk202011\edk2\Build\EmulatorIA32\DEBUG_VS2015x86\IA32.

4) Configure the simulator network environment

Copy the previously compiled SnpNt32Io.dll to the directory where the simulator is located, and create a batch file loadnetwork.nsh in this directory with the following content:

```
c                                                                    AI generated projects  登录复制  run
1 | load MnpDxe.efi ArpDxe.efi Ip4Dxe.efi VlanConfigDxe.efi Udp4Dxe.efi Dhcp4Dxe.efi Mtftp4Dxe.efi TcpDxe.efi
```

The UEFI drivers in the batch file can be found in the simulator directory and are compiled together when the simulator is compiled.

Double-click the executable file WinHost.exe in the simulator directory to start the simulator and enter the UEFI Shell environment. Run the batch file to load network-related drivers:

cAI generated projects登录复制run

```
1 | Shell> fs0:
2 | FS0:\> loadnetwork.nsh
```

The IP address of my machine is 192.168.181.132, and the gateway is 192.168.181.2. Set the network address in the simulator to 192.168.181.135 with the following command:

cAI generated projects登录复制run

```
1 | FS0:\> ifconfig -s eth0 static 192.168.181.135 255.255.255.0 192.168.181.2
```

After the configuration is complete, you can use the "ifconfig -l eth0" and ping commands to check whether the configuration is successful.

cAI generated projects登录复制run

```
1 | FS0:\> ifconfig -l eth0
2 | name      : eth0
3 | Media State : Media state unknown
4 | policy     : static
5 | mac addr   : 00:0C:29:D5:97:D7
6 | ipv4 address : 192.168.181.135
7 | subnet mask : 255.255.255.0
8 | default gateway: 192.168.181.2
9 |
10 | Routes(2 entries):
11 |   Entry[0]
12 | Subnet  : 192.168.181.0
13 | Netmask : 255.255.255.0
14 | Gateway: 0.0.0.0
15 |   Entry[1]
16 | Subnet  : 0.0.0.0
17 | Netmask : 0.0.0.0
18 | Gateway: 192.168.181.2
19 | DNS server :
20 |
twen | FS0:\> ping 192.168.1.42
twen | ping 192.168.1.42 16data bytes.
twen | 16 bytes from 192.168.1.42 : icmp_seq=1 ttl=0 time0~26ms
twen | ...
```

收起 ^

My UEFI development machine is a VMware virtual machine with Win10 operating system, on which the UEFI development environment is built. The host machine's operating system is also Win10, and a bunch of virtual machines are running on the host machine for development of various needs.

The host machine is equivalent to another machine in the local area network, and its IP address is 192.168.1.42.

In short, the network environment has been set up, and the network program test is now carried out.

3 Network Program Testing

Previously, in order to test the UEFI network, a server program on the Windows side and a client program on the UEFI side were developed, namely EchoServerTCP4.exe and EchoTcp4.efi.

The EchoServerTcp4.exe program has been compiled before, and only the client code needs to be compiled.

Copy RobinPkg from the repository of the UEFI Development Exploration Blog to the currently built UEFI development environment, and compile EchoTcp4.efi:

c

AI generated projects

登录复制

run

1 | C:\edk202011\edk2>build -p RobinPkg\RobinPkg.dsc -m RobinPkg\Applications\EchoTcp4\EchoTcp4.inf -a IA32

(I recently reworked EchoTcp4 and updated the code. In fact, the same is true for projects using stdEchoTcp4.)

Copy the compiled EchoTcp4.efi to the directory where the simulator is located.

Run the server program on the host machine and the client program on the EDK2 simulator. The test results are as follows:

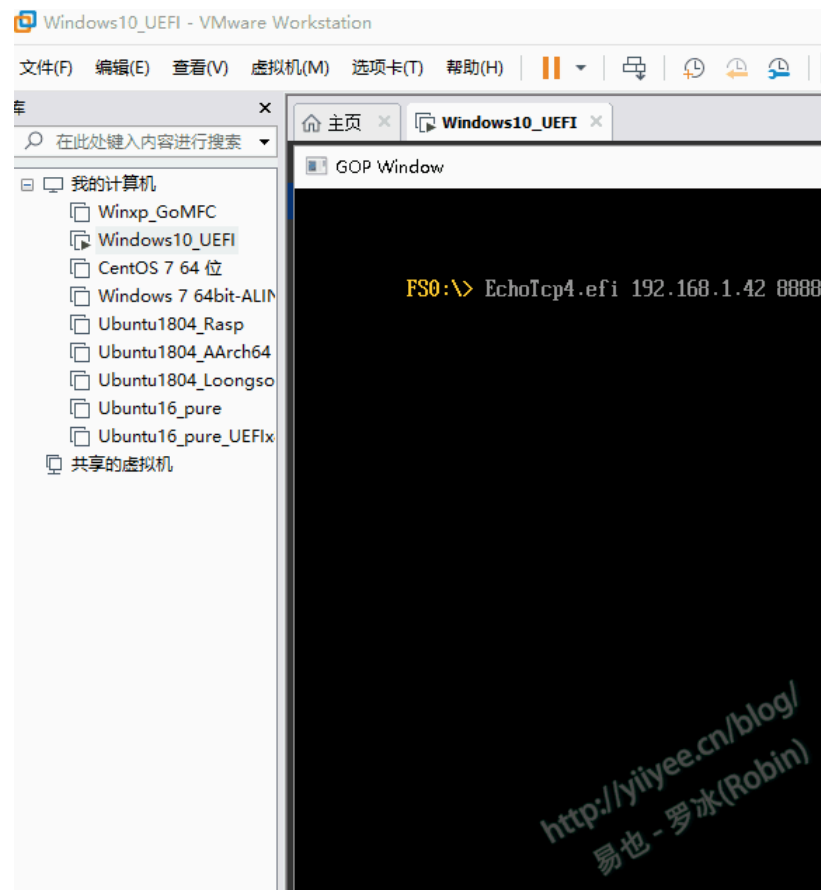


Figure 2 Test network communication

Of course, you can also use the commonly used network debugging assistant to test. The test results are as follows:

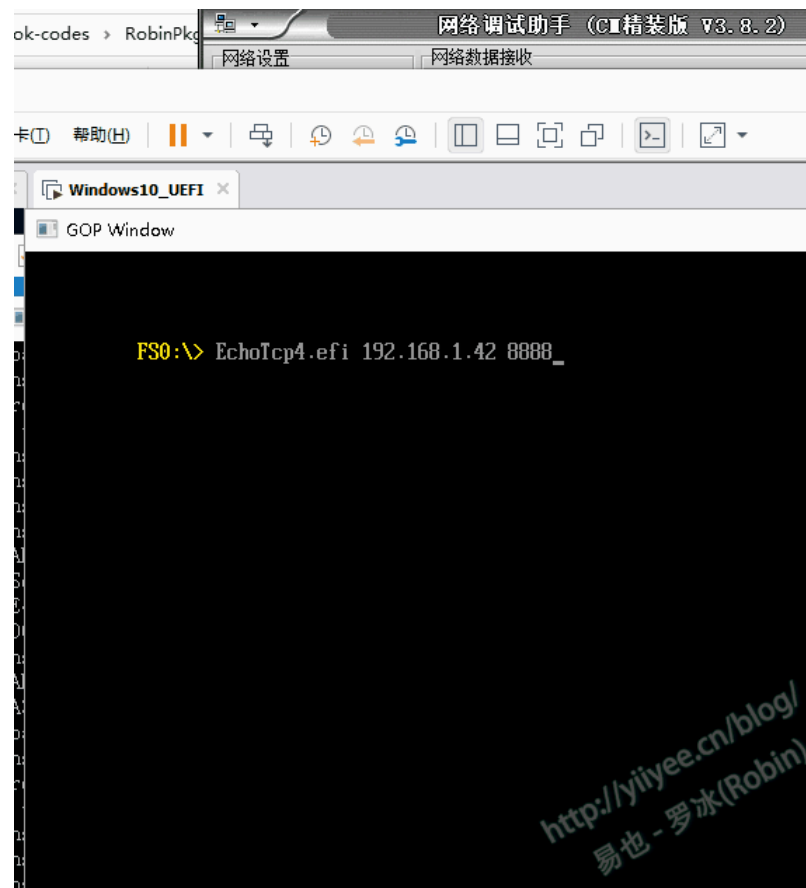


Figure 3 Testing with the network debugging assistant

Normally I rarely need to debug UEFI network programs, so I always use the environment built by the method in this article for testing.

During the experiment, I also found several problems:

- 1) The firewall will affect the sending of packets, so it is best to turn off all firewalls. Maybe Winpcap's packet capture ability is affected. I have never used Winpcap to write programs. Netizens who understand the principle can help me take a look;
- 2) When setting the EDK2 simulator URL, when setting it to DHCP, sometimes it succeeds, sometimes it fails, I suspect it is related to the router. The specific reason is unknown, maybe this problem can be explained after deepening the simulator principle.
- 3) On the operating system running the EDK2 simulator, it is unsuccessful to use the server software to communicate with the client software on the simulator. In other words, the server software needs to be run on other machines in the same LAN. For example, I use a host machine to run the server software, a virtual machine Win10 to run the EDK2 simulator, and the client software on the simulator runs normally.

These questions are for your reference. I am not sure about the specific reasons. You are welcome to leave a message for discussion.

To facilitate the experiment, I put the source code of Winpcap, WdpPack (Winpcap development kit) and NetNt32Io required for this experiment in the warehouse:

Gitee address: <https://gitee.com/luobing4365/uefi-explorer>

Project files are located under: /97 Network-Emulator

关于我们 招贤纳士 商务合作 寻求报道  400-660-0108  kefu@csdn.net  在线客服 Working hours 8:30-22:00

Public Security Registration Number 11010502030143 Beijing ICP No. 19004658 Beijing Internet Publishing House [2020] No. 1039-165

Commercial website registration information Beijing Internet Illegal and Harmful Information Reporting Center Parental Control

Online 110 Alarm Service China Internet Reporting Center Chrome Store Download Account Management Specifications

Copyright and Disclaimer Copyright Complaints Publication License Business license

©1999-2025 Beijing Innovation Lezhi Network Technology Co., Ltd.