[UEFI Practice] Write your own Shell command

jiangwei0512 Posted on 2016-09-05 14:20:51 Read 2w Collection 29 Likes 6

Category Column: UEFI Development Basics Article Tags: uefi shell

摘要 This article introduces how to add custom commands in UEFI Shell Ver2 by modifying the ShellPkg source code and compiling it into OVMF. The specific steps include creating lib and inf files, writing command functions, etc.

The summary is generated in C Know , supported by DeepSeek-R1 full version, go to experience>

illustrate

This article writes a Shell command based on UDK2015 and runs it under Shell.

There are two versions of Shell under UEFI, one is Shell Ver 1, corresponding to EdkShellPkg; the other is Shell Ver 2, corresponding to ShellPkg.

Currently, there is no source code for EdkShellPkg in UDK2015 and it needs to be downloaded separately.

So this article is based on the source code in ShellPkg.

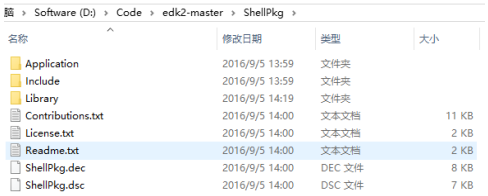
There is no way to run a Shell directly. In this article, Shell is attached to OVMF, so OvmfPkgX64.dsc is used for compilation.

The advantage of using OVMF is that it can be run through qemu. Another advantage is that OvmfPkgX64.dsc already contains ShellPKg.dsc, so no additional operations are required.

For the compilation of OVMF, please refer to [UEFI Practice] OVMF Basics .

Add source code

As mentioned earlier, the source code of Shell is located in the ShellPkg directory:

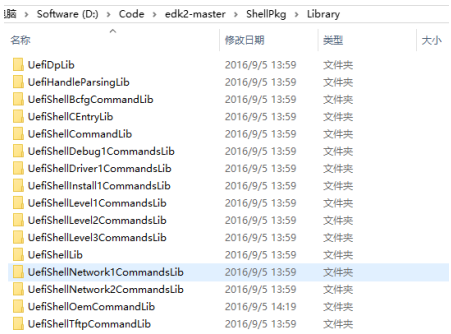


in:

Application contains Shell itself and some simple application examples. These applications, including Shell itself, can be run directly under Shell.

Include contains some required header files.

Library contains the basic libraries required by Shell and the commands that can be executed under Shell:

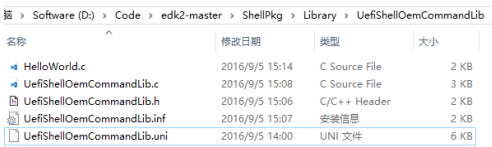


In Shell Ver 2, Shell commands are included in the library. For example, the UefiShellNetwork1CommandsLib above contains the ifconfig and ping commands.

In Shell Ver 2, each command is divided into different directories according to its function.

This article will create its own Lib in the above format and implement commands in it.

Create OemLib



What needs to be explained here are the inf and uni files:

inf is used for compilation and represents a module.

uni is a string file used to display some help commands or error messages in Shell commands.

After that you need to add the inf file to ShellPkg.dsc:

```
!ifndef $(USE_OLD_SHELL)
  ShellPkg/Application/Shell/Shell.inf {
    <LibraryClasses>
      ShellCommandLib|ShellPkg/Library/UefiShellCommandLib/UefiShellCommandLib.inf
      NULL|ShellPkg/Library/UefiShellLevel2CommandsLib/UefiShellLevel2CommandsLib.inf
      NULL|ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.inf
      NULL|ShellPkg/Library/UefiShellLevel3CommandsLib/UefiShellLevel3CommandsLib.inf
      NULL|ShellPkg/Library/UefiShellDriver1CommandsLib/UefiShellDriver1CommandsLib.inf
      NULL|ShellPkg/Library/UefiShellDebug1CommandsLib/UefiShellDebug1CommandsLib.inf
      NULL|ShellPkg/Library/UefiShellInstall1CommandsLib/UefiShellInstall1CommandsLib.inf
      NULL|ShellPkg/Library/UefiShellNetwork1CommandsLib/UefiShellNetwork1CommandsLib.inf
      ## jw_debug
      NULL|ShellPkg/Library/UefiShellOemCommandLib/UefiShellOemCommandLib.inf
      ## jw_debug
!if $(NETWORK_IP6_ENABLE) == TRUE
      NULL|ShellPkg/Library/UefiShellNetwork2CommandsLib/UefiShellNetwork2CommandsLib.inf
!endif
      NULL|ShellPkg/Library/UefiShellTftpCommandLib/UefiShellTftpCommandLib.inf
      HandleParsingLib|ShellPkg/Library/UefiHandleParsingLib/UefiHandleParsingLib.inf
      ShellLib|ShellPkg/Library/UefiShellLib/UefiShellLib.inf
      FileHandleLib|MdePkg/Library/UefiFileHandleLib/UefiFileHandleLib.inf
      PrintLib|MdePkg/Library/BasePrintLib/BasePrintLib.inf
#       SafeBlockIoLib|ShellPkg/Library/SafeBlockIoLib/SafeBlockIoLib.inf
#       SafeOpenProtocolLib|ShellPkg/Library/SafeOpenProtocolLib/SafeOpenProtocolLib.inf
      BcfgCommandLib|ShellPkg/Library/UefiShellBcfgCommandLib/UefiShellBcfgCommandLib.inf

    <PcdsFixedAtBuild>
      gEfiMdePkgTokenSpaceGuid.PcdDebugPropertyMask|0xFF
      gEfiShellPkgTokenSpaceGuid.PcdShellLibAutoInitialize|FALSE
      gEfiMdePkgTokenSpaceGuid.PcdUefiLibMaxPrintBufferSize|8000
  }
!endif
```

This is how it can be compiled into OVMF.

Specific code

cpp — AI generated projects — 登录复制 — run

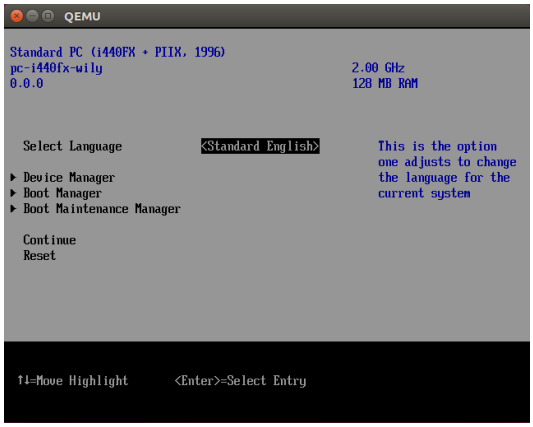
```cpp
#include "UefiShellOemCommandLib.h"

CONST CHAR16 gShellOemFileName[] = L"ShellCommand";
EFI_HANDLE gShellOemHiiHandle = NULL;

/**
  Return the file name of the help text file if not using HII.

  @return The string pointer to the file name.
**/
CONST CHAR16*
EFIAPI
ShellCommandGetManFileNameOem (
  VOID
  )
{
  return gShellOemFileName;
}

/**
  Constructor for the Shell xxx Command library.

  Install the handlers for xxx UEFI Shell command.

  @param ImageHandle            The image handle of the process.
  @param SystemTable            The EFI System Table pointer.

  @retval EFI_SUCCESS           The Shell command handlers were installed sucessfully.
  @retval EFI_UNSUPPORTED       The Shell level required was not found.
**/
EFI_STATUS
EFIAPI
ShellOemCommandLibConstructor (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
  gShellOemHiiHandle = NULL;

  //
  // check our bit of the profiles mask
  //
  if ((PcdGet8 (PcdShellProfileMask) & BIT3) == 0) {
    return EFI_SUCCESS;
  }

  gShellOemHiiHandle = HiiAddPackages (
                         &gShellOemHiiGuid, gImageHandle,          // gShellOemHiiGuid需要在ShellLibHiiGuid.h和ShellPkg.dec中定义,并声明在UefiShellOemCommandLib.inf
                         UefiShellOemCommandLibStrings, NULL       // UefiShellOemCommandLibStrings就对应到UefiShellOemCommandLib.uni
                         );
  if (gShellOemHiiHandle == NULL) {
    return EFI_DEVICE_ERROR;
  }
  //
  // Install our Shell command handler
  //
  ShellCommandRegisterCommandName (
     L"helloworld", ShellCommandRunHelloWorld, ShellCommandGetManFileNameOem, 0,
     L"helloworld", TRUE , gShellOemHiiHandle, STRING_TOKEN (STR_GET_HELP_OEM)  // STR_GET_HELP_OEM在UefiShellOemCommandLib.uni中定义
     );

  return EFI_SUCCESS;
}

/**
  Destructor for the library.  free any resources.

  @param ImageHandle            The image handle of the process.
  @param SystemTable            The EFI System Table pointer.
**/
EFI_STATUS
EFIAPI
ShellOemCommandLibDestructor (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
  if (gShellOemHiiHandle != NULL) {
    HiiRemovePackages (gShellOemHiiHandle);
  }
  return EFI_SUCCESS;
}
```

收起 ^

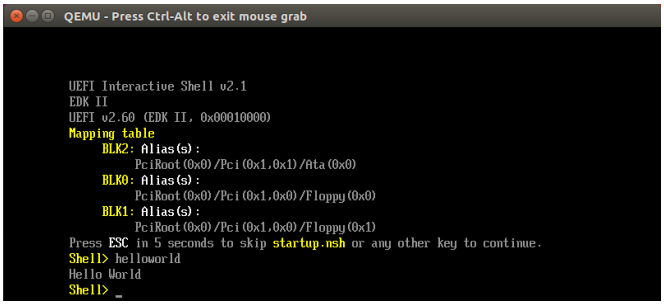
The codes of other files are omitted.

Operation Results

Run qemu, press the key after opening the qemu window, and the UEFI Front Page will be displayed.



Select Boot Manager, enter Shell, and run helloworld:



The above examples can be found in the git repository https://code.csdn.net/jiangwei0512/bios_git.git. The specific code may be slightly different.

Update 20180614:

The code has been updated to https://gitee.com/jiangwei0512/vUDK2017.

For details, see the code in the ShellPkg\Library\UefiShellBeniCommandLib\ directory.