

UEFI Development Exploration 46 – UEFI Support C++

原创

luobing4365

Posted on 2020-03-02 21:34:02

Read 1.4k

Collection 1

Likes 1

copyright

Category Column: UEFI Development

Article Tags: UEFI Programming

UEFI C++

EDK2

Low-level programming

Low-level application development



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>)

Since GuiLite was developed in C++, porting it to UEFI means that the code must be written in C++.

Most of the following content comes from Chapter 10 of UEFI Principles and Programming. I integrated the required code into my own **framework** . Some small problems that are not encountered in the book are also solved in the blog.

1 C and C++ for UEFI

Programs in UEFI follow the rules of C language, and most of the time you should use modular thinking to understand the code. There is no problem in daily development, and C language is more suitable for dealing with hardware directly than C++. However, when encountering a slightly larger **GUI** application or porting legacy C++ code, you have to use C++. For example, this time's GuiLite porting work.

In **Windows** , the development tool is Visual Studio, and **the compiler** cl itself supports C++. Therefore, what we need to do is to make the UEFI entry function "recognize" C++ code.

The main difference between C++ and C is that function names, array names, etc. will be different after compilation. This is the process of name decoration, which is described in most books on compiler theory. Therefore, if you use C code directly to call C++ libraries, the link will fail.

In order to support C++, EDK2 has gradually added some support in subsequent versions. For example, in the generated intermediate file **AutoGen.h** , extern "C" is automatically added to solve the name modification problem of the intermediate file.

My development environment is UDK2018+VS2015, and I develop on Win10.

2 Supporting basic classes

Basically, the following four steps can support the functions of C++ classes.

1) Solve the name modification problem

This problem is relatively simple. You only need to include the C language header file with extern "C" in the C++ source file. As shown in the figure:

```
extern "C"
{
#include <Uefi.h>
#include <Library/UefiLib.h>
#include <Library/ShellCEntryLib.h>
#include <Library/DebugLib.h>

#include <Library/BaseMemoryLib.h>

#include <Library/UefiBootServicesTableLib.h>
#include <Library/BaseLib.h>
// #include <Protocol/SimpleTextInEx.h>

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <wchar.h>
}
```

Figure 1 Name decoration problem between C and C++

The name decoration problem of AutoGen.h in the old version of EDK2 is not solved. The version I am using now is close to it and can be ignored.

2) Using NULL in C++

To use NULL in UEFI C++ code, you need to redefine:

```
#undef NULL
#define NULL 0
```

3) Using Boolean in C++

You can modify the compilation options in the inf file to solve this problem:

```
[Protocols]
gEfiSimpleTextInputExProtocolGuid          ## CONSUMES
[BuildOptions]
# MSFT:*_*_IA32_CC_FLAGS = /O1- /wd4804
MSFT:*_*_CC_FLAGS = /wd4804
```

Figure 2 Solving the problem of using Boolean variables

4) /Zc:wchar_t problem

My sample code uses functions from StdLib. When compiling, I get this error:

```
"C:\Program Files (x86)\Microsoft Visual Studio 14.0\bin\cl.exe" /Foc:\myworkspace\Build\RobinPkg\DEB
UG_VS2015x86\IA32\RobinPkg\Applications\CppMain\OUTPUT\CppMain.obj /nologo /arch:IA32 /c /WX /GS- /W4
/Gs32768 /D UNICODE /FIAutoGen.h /EHs-c- /GR- /GF /Gy /Zi /Gm /Gw /wd4804 /X /Zc:wchar_t /D UEFI_C_SOURCE /O1:11
/Ic:\myworkspace\RobinPkg\Applications\CppMain /Ic:\myworkspace\Build\RobinPkg\DEBUG_VS2015x86\IA32\RobinPkg\A
pplications\CppMain\CppMain\DEBUG /Ic:\myworkspace\StdLib /Ic:\myworkspace\StdLib\Include /Ic:\myworkspace\St
dLib\Include\ia32 /Ic:\myworkspace\MdePkg /Ic:\myworkspace\MdePkg\Include /Ic:\myworkspace\UefiPkg\Include\ia32
2 /Ic:\myworkspace\ShellPkg /Ic:\myworkspace\ShellPkg\Include c:\myworkspace\RobinPkg\Applications\CppMain\Cpp
Main.cpp
CppMain.cpp
c:\myworkspace\StdLib\Include\sys\EfiCdefs.h(330): fatal error C1189: #error: You must specify /Zc:wchar_t- to
the compiler to turn off intrinsic wchar_t.
NMAKE : fatal error U1077: "C:\Program Files (x86)\Microsoft Visual Studio 14.0\bin\cl.exe" : 返回代码 "0x
2"
```

Figure 3 Wide character type error

Simply comment out the statement on line 330 in EfiCdefs.h.

```
329 #ifdef _NATIVE_WCHAR_T_DEFINED
330 // #error You must specify /Zc:wchar_t- to the compiler to turn off intrinsic wchar_t.
331 #endif
```

Figure 4 Rewriting of EfiCdefs.h

After the above four steps, the UEFI code can support basic classes, including class **constructors**, destructors, and derivations. However, if you need to support global instantiation of classes, you still need to do some work.

Global instantiation of 3 classes

Here I directly borrowed the code from "UEFI Principles and Programming".

Specifically, we can observe how C++ performs global construction and destruction from the assembly language, imitate its process, and process global class instances before and after executing the actual application code.

This method is relatively complicated, please refer to the original book for details. It should be noted that this is the processing method of Visual Studio on the Windows platform. If you change the platform or compiler, the code will not be universal.

This is the constructed code framework:

```
> NetworkPkg
> Nt32Pkg
> Omap35xxPkg
> OptionRomPkg
> OvmfPkg
> PcAtChipsetPkg
> QuarkPlatformPkg
> QuarkSocPkg
> RobinPkg
  > Applications
    > CppMain
      > Cppglobal.h
      > CppMain.cpp
      > CppMain.inf
      > crt0data.cpp
    > Luo3
      > Luo3.c
      > Luo3.inf
    > RngEvent
      > Common.c
      > Common.h
      > FileRW.c
      > FileRW.h
      > Graphic.c
      > Graphic.h
      > Keyboard.c
  > OUTLINE
    > testClass
      39 class testClass
      40 { ...
      50 };
      51 EFI_STATUS GetKeyEx(UINT16 *ScanCode, UINT16 *UniChar, UINT32 *Shift
      52
      53 testClass t2;
      54 > /***...
      67 ***/
      68 int
      69 > main ( ...
      73 {
      74 // EFI_STATUS Status;
      75 // UINT16 scanCode=0;
      76 // UINT16 uniChar=0;
      77 // UINT32 shiftState;
      78 // EFI_KEY_TOGGLE_STATE toggleState;
      79 // UINT32 count=0;
      80 __do_global_ctors_aux();
      81
      82 gST->ConOut->OutputString(gST->ConOut,(CHAR16 *)L"===== EDKI
      83 gST->ConOut->OutputString(gST->ConOut,(CHAR16 *)L"Author: luobin
      84 gST->ConOut->OutputString(gST->ConOut,(CHAR16 *)L"Data: 2020-3-1
      85 gST->ConOut->OutputString(gST->ConOut,(CHAR16 *)L"Context: uefi
      86 gST->ConOut->OutputString(gST->ConOut,(CHAR16 *)L"=====
      87 //text out test
      88
      89 // testClass t;
      90 Print((CHAR16 *)L"This is a sample of uefi(c++)!");
      91
      92 __do_global_dtors_aux();
      93 return 0;
      94 }
```

Figure 5 UEFI C++ framework code

Among them, crt0data.cpp comes from the code in the above reference book. In order to indicate the source, I also retained the name of the source file.

At this point, the framework is complete. As for the memory handling method that supports new and delete, it is not used in GUI Lite, so I will not add it.

Gitee address: <https://gitee.com/luobing4365/uefi-explorer>
Project code is located at: / FF RobinPkg/RobinPkg/Applications/CppMain

无数C++开发者追捧的经典

广告


所谓吾道一以贯之，胸中自有丘壑。侯捷大师亲授的C++系列教程全面覆盖C++核心技术，从面向对象到C++ 2.0新特性，用实战案例打通开发瓶颈。


about
Us


Careers

Business
Cooperation

Seeking
coverage

 400-660-0108

 kefu@csdn.net

 Online
Customer
Service

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.