







# UEFI Basic Tutorial (Ten) - Simple Use of FileIO

 xiaopangzi313  Posted on 2020-06-06 10:23:56  Read 3.5k  Collection 4  Likes 1

Copyright CC 4.0 BY-SA


Category columns: 

15\_Firmware Development

 15\_Firmware Devel... This column includes this content

27 articles 

Subscribe to our column

 **摘要**

This article demonstrates how to use EFI\_SHELL\_PROTOCOL to read, copy and output files in the UEFI environment. It also introduces the source code writing, compilation process and the process of running UEFI applications in a virtual environment.

The summary is generated in [C Know](#) , supported by DeepSeek-R1 full version, [go to experience>](#)

## I. Write source code

1. Write the UEFI Application
- code C:\edkii\OvmfPkg\MyHelloWorldFileIO\MyHelloWorldFileIO.c,

AI generated projects 登录复制

```
1  EFI_STATUS OpenShellProtocol( EFI_SHELL_PROTOCOL **gEfiShellProtocol )
2  {
3      EFI_STATUS Status;
4      Status = gBS->OpenProtocol(
5          gImageHandle,
6          &gEfiShellProtocolGuid,
7          (VOID **)gEfiShellProtocol,
8          gImageHandle,
9          NULL,
10         EFI_OPEN_PROTOCOL_GET_PROTOCOL
11     );
12     if (EFI_ERROR(Status)) {
13         //
14         // Search for the shell protocol
15         //
16         Status = gBS->LocateProtocol(
17             &gEfiShellProtocolGuid,
18             NULL,
19             (VOID **)gEfiShellProtocol
20         );
21     }
22     if (EFI_ERROR(Status)) {
23         gEfiShellProtocol = NULL;
24     }
25     return Status;
26 }
27
28 INTN
29 EFIAPI
30 ShellAppMain( UINTN Argc, CHAR16 **Argv)
31 {
32     CHAR16 * OldLogFileName = NULL;
33     CHAR16 * LineBuff = NULL;
34     CHAR16 NewFileName[128] = {0};
35     CHAR16 * ArrayBuffer = NULL;
36     EFI_STATUS Status ;
37     SHELL_FILE_HANDLE FileHandle;
```

```

38  UINTN Index = 0;
39  UINTN WbufSize = 0;
40  UINTN FileSize = 0;
41  UINTN i = 0;
42  UINTN StartIndex = 0;
43  CHAR8 *Ptr = NULL;
44
45  if (Argc <= 1){
46      Print(L"Please input file name!\n");
47      return (-1);
48  }
49  OldLogFileName = Argv[1];
50  Print(L"The Old file name is %s!\n",OldLogFileName);
51
52  EFI_SHELL_PROTOCOL *gEfiShellProtocol;
53  Status = OpenShellProtocol(&gEfiShellProtocol);
54  Status = gEfiShellProtocol->OpenFileByName((CONST CHAR16*)OldLogFileName, &FileHandle, EFI_FILE_MODE_READ);
55  if (EFI_ERROR(Status)){
56      Print(L"Please Input Valid Filename!\n");
57      return (-1);
58  }
59  StrnCpyS(NewFileName,128,OldLogFileName,StrLen(OldLogFileName)-4);
60  StrCatS(NewFileName,128,L"_New.txt");
61  Print(L"New FileName is %s\n",NewFileName);
62
63  //删除同名的文件
64  gEfiShellProtocol->DeleteFileByName(NewFileName);
65
66  //获取文件实际大小
67  Status = gEfiShellProtocol->GetFileSize(FileHandle,&FileSize);
68  if (EFI_ERROR (Status)) {
69      gEfiShellProtocol->CloseFile (FileHandle);
70      return Status;
71  }
72  Print (L"File FileSize is %d!\n",FileSize);
73
74  if (FileSize < 0){
75      Print (L"File cotent is empty!\n");
76      return (-1);
77  }
78
79  FileSize += 1;
80
81  //根据文件大小申请对应大小的内存
82  Status = gBS -> AllocatePool (EfiReservedMemoryType, FileSize , &ArrayBuffer);
83  Status = gBS -> AllocatePool (EfiReservedMemoryType, FileSize , &LineBuff);
84
85  BZero(ArrayBuffer,FileSize);
86  BZero(LineBuff,FileSize);
87  Status = gEfiShellProtocol->ReadFile(FileHandle, &FileSize ,ArrayBuffer);
88
89  if (EFI_ERROR(Status)){
90      Print(L"Read Filename Error!\n");
91      return (-1);
92  }
93
94  //创建新的文件句柄
95  Status = gEfiShellProtocol->CreateFile((CONST CHAR16*)NewFileName,0 ,&FileHandle);
96  if (EFI_ERROR(Status)){
97      Print(L"Create Filename %s Fail!\n",NewFileName);
98      return (-1);
99  }
100
101  //读取的文件内容写入新建文件
102  WbufSize = FileSize;
103

```

```

104 | Status = gEfiShellProtocol->WriteFile(FileHandle,&WbufSize,ArrayBuffer);
105 |
106 | //关闭文件句柄
107 | Status = gEfiShellProtocol->CloseFile(FileHandle);
108 |
109 | Ptr = (CHAR8 * )ArrayBuffer;
110 | for (i = 0 ; i < FileSize ; i ++ ){
111 |     if (Ptr[i] == '\n'){
112 |         Ptr[i] = '\0';
113 |         Ascii2UnicodeString(Ptr + StartIndex, LineBuff);
114 |         StartIndex = i + 1;
115 |
116 |         Index += 1;
117 |         //按行输出
118 |         Print(L"Line %d: %S!\n",Index,LineBuff);
119 |         BZero(LineBuff,FileSize);
120 |     }
121 | }
122 |
123 | return (0);
}

```

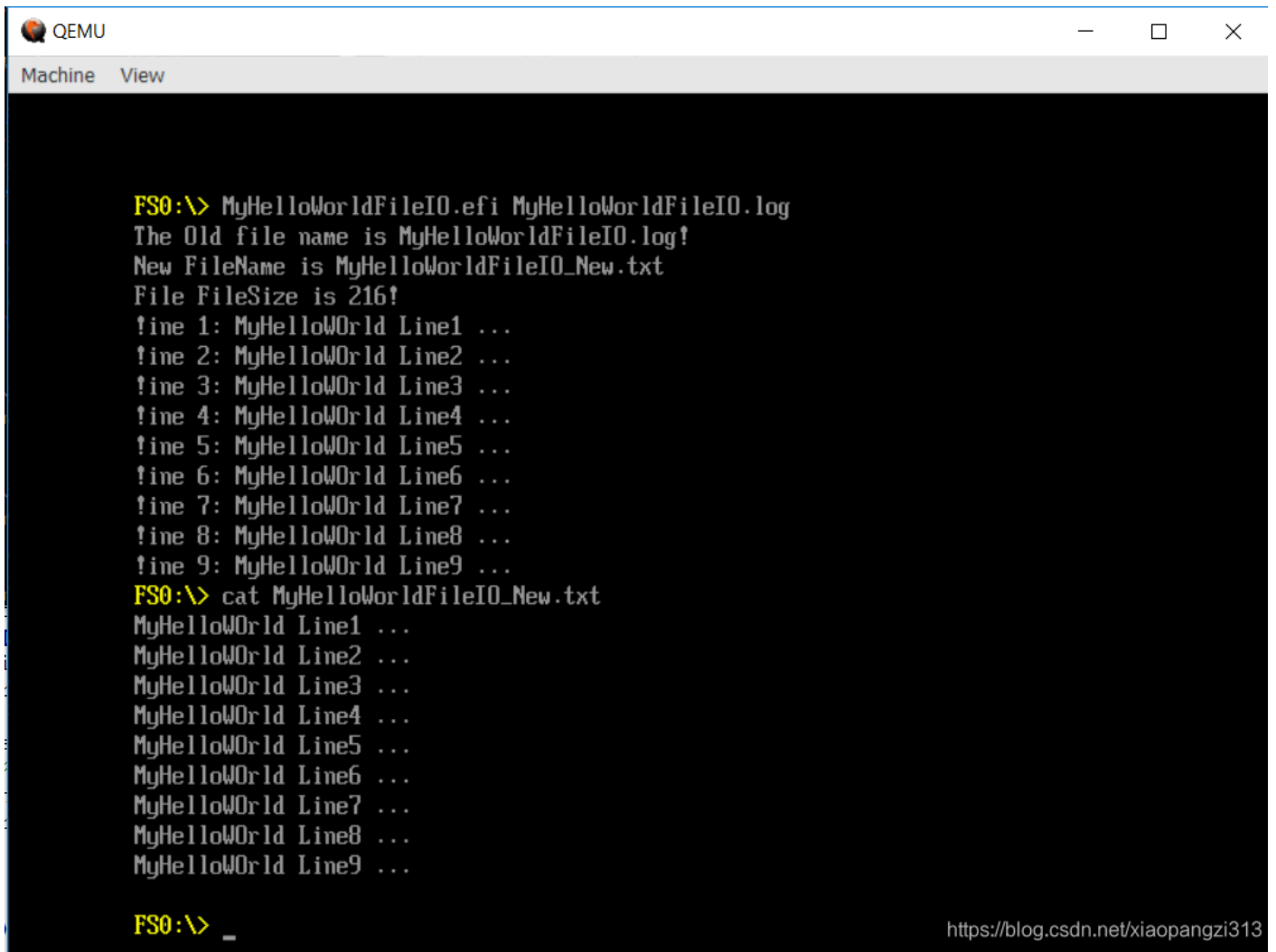
收起 ^

## 2. Compile and generate EFI files

Run and **edksetup.bat** compile the entire OvmfPkg Package

## 3. Run UEFI APP **MyHelloWorldFileIO.efi**

1. Copy **C:\edkii\Build\OvmfX64\DEBUG\_VS2013x86\FV\OVMF.fd** to **C:\qemu** ; Copy **C:\edkii\Build\OvmfX64\DEBUG\_VS2013x86\X64\OvmfPkg\MyHelloWorldFileIO\MyHelloWorldFileIO\OUTPUT\MyHelloWorldFileIO.efi** to virtual disk **HDD\_BOOT.img**
2. Execute **etup-qemu-x64.bat** , then **UEFI SHELL** execute in **MyHelloWorldFileIO.efi** , the result is as follows,  
[External link image transfer failed, the source site may have an anti-hotlink mechanism, it is recommended to save the image and upload it directly



The image shows a QEMU terminal window with a black background and yellow text. The terminal displays a series of commands and their outputs. The first command is `FS0:\> MyHelloWorldFileIO.efi MyHelloWorldFileIO.log`, which outputs: `The Old file name is MyHelloWorldFileIO.log!`, `New FileName is MyHelloWorldFileIO_New.txt`, and `File FileSize is 216!`. This is followed by a loop of commands `!ine 1: MyHelloWOrld Line1 ...` through `!ine 9: MyHelloWOrld Line9 ...`. The next command is `FS0:\> cat MyHelloWorldFileIO_New.txt`, which outputs the same nine lines. The prompt `FS0:\> _` is shown at the bottom left. A URL `https://blog.csdn.net/xiaopangzi313` is visible in the bottom right corner of the terminal window.

```
FS0:\> MyHelloWorldFileIO.efi MyHelloWorldFileIO.log
The Old file name is MyHelloWorldFileIO.log!
New FileName is MyHelloWorldFileIO_New.txt
File FileSize is 216!
!ine 1: MyHelloWOrld Line1 ...
!ine 2: MyHelloWOrld Line2 ...
!ine 3: MyHelloWOrld Line3 ...
!ine 4: MyHelloWOrld Line4 ...
!ine 5: MyHelloWOrld Line5 ...
!ine 6: MyHelloWOrld Line6 ...
!ine 7: MyHelloWOrld Line7 ...
!ine 8: MyHelloWOrld Line8 ...
!ine 9: MyHelloWOrld Line9 ...
FS0:\> cat MyHelloWorldFileIO_New.txt
MyHelloWOrld Line1 ...
MyHelloWOrld Line2 ...
MyHelloWOrld Line3 ...
MyHelloWOrld Line4 ...
MyHelloWOrld Line5 ...
MyHelloWOrld Line6 ...
MyHelloWOrld Line7 ...
MyHelloWOrld Line8 ...
MyHelloWOrld Line9 ...

FS0:\> _
```

<https://blog.csdn.net/xiaopangzi313>

## V. Summary

File operation is an important function in UEFI. This article uses `EFI_SHELL_PROTOCOL` to read files, write new files, and then output them to the Console line by line. It should be noted that if you read ASCII files, the output terminal needs to be able to convert between ASCII and Unicode.

## FileIO DEMO source code

## 金角大王Alex破界！技术流TikTok掘金

广告

首次揭秘：用Python脚本收割欧美流量池→ 工程师的跨境第一课