UEFI Principles and Programming Practice--Device Path

原创 Anthony Posted on 2021-11-29 13:43:55 Read 2.4k Collection 12 Likes 4

Category Column: UEFI Article Tags: UEFI

摘要 This article describes how to convert a device path to a string using the EFI_DEVICE_PATH_TO_TEXT_PROTOCOL protocol in a UEFI environment, and how to traverse the device path nodes. The sample code shows the process of finding and printing all hard disk device paths that support the DiskIo protocol.

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

Each device in the system has a unique path. For example, every time you enter the shell, the hard disk device and device path in the system will be printed out. For the hard disk and file system, in the BIOS learning practice, we will obtain the USB disk path, read the BIOS file, and then perform the specific operation of updating. This article mainly talks about some basic content.

Basics

The nodes in a device path are called device nodes. A device path consists of a list of device nodes, and the list ends with an end device node.

Each device node starts with EFI_DEVICE_PATH_PROTOCOL. The end device node is a special device node. Its type is 0x7F, the subtype is 0xFF or 0x01, and the node length is 2 bytes.

字节偏移	域 名 称	值	值的意义
0	类型	0x7F	结束标志
1	次类型	0xFF/0x01	0xFF：整个设备路径结束 0x01：前一设备路径结束；新的设备路径开始
2	节点长度	4	结束节点没有数据，因而长度为 4 字节
3			

表 7-6 硬盘设备节点

域	偏移	域长度	域类型	域的含义
Type	0	1	UINT8	介质设备路径，值为 0x4
SubType	1	1	UINT8	硬盘，值为 1
Length	2	2	UINT16	长度，大小为 42 字节
PartitionNumber	4	4	UINT32	该分区在分区表中的位置，从 1 开始编号。 0 表示整个硬盘设备。 对 MBR 硬盘来说，有效值是 1、2、3、4 中的一个；对 GPT 硬盘来说，有效值的范围是 [1，分区个数]
PartitionStart	8	8	UINT64	该分区第一个扇区的 LBA 地址
PartitionSize	16	8	UINT64	该分区的扇区数
Signature	24	16	UINT8[16]	该分区的标识符。 如果 SignatureType 为 0，本域必须为 0；如果 SignatureType 为 1，本域前 4 字节有效，后面 12 字节为 0；如果 SignatureType 为 2，本域必须是一个有效的 GUID
MBRType	40	1	UINT8	1 表示 MBR 硬盘；2 表示 GPT 硬盘
SignatureType	41	1	UINT8	分区标识符的类型，0 表示无分区标识符；1 表示 MBR 分区（32 位）的标识符；2 表示标识符为 GUID

Here are the key points:

UEFI provides EFI_DEVICE_PATH_TO_TEXT_PROTOCOL for converting a device path to a string, in which the member function ConvertDevicePathToText is used to convert a device path DevicePath to a string.

IsDevicePathEnd (CONST VOID *Node) is used to determine whether the device node Node is the device end node of the device path.

NextDevicePathNode (CONST VOID *Node) is used to return the next device node of the device node Node.

Taking printing the found hard disk device path as an example, the steps are as follows:

1. First, use the gBS->LocateHandleBuffer service to find all devices that support DiskIo.

2. Then find the device path of the DiskIo device

3. Call ConvertDevicePathToText to get the device path string

4. Traverse each node of the device path

The code is as follows:

```
#include <DevicePath.h>


EFI_STATUS

PrintNode(EFI_DEVICE_PATH_PROTOCOL *Node){

     Print(L"(%d %d)/",Node->Type,Node->SubType);

     return 0;

}


EFI_DEVICE_PATH_PROTOCOL*

WalkthroughDevicePath(

   EFI_DEVICE_PATH_PROTOCOL* DevPath,

   EFI_STATUS (*Callbk)(EFI_DEVICE_PATH_PROTOCOL*)

   )

{

     EFI_DEVICE_PATH_PROTOCOL* pDevPath=DevPath;

     while(!IsDevicePathEnd(pDevPath)){

        Callbk(pDevPath);

        pDevPath= NextDevicePathNode(pDevPath);

     }

     return pDevPath;

}




EFI_STATUS

EFIAPI

ShellAppMain (

  IN UINTN Argc,

  IN CHAR16 **Argv

  )

{

  EFI_STATUS          Status ;

  UINTN               HandleIndex,NumHandles;

  EFI_HANDLE *ControllerHandle =NULL;

  EFI_DEVICE_PATH_TO_TEXT_PROTOCOL *Device2TextProtocol = 0;

  EFI_DEVICE_PATH_PROTOCOL *DiskDevicePath;

  CHAR16 *TextDevicePath;


 //第一步，打开EFI_DEVICE_PATH_TO_TEXT_PROTOCOL服务
  Status = gBS->LocateProtocol(
```

```
                        &gEfiDevicePathToTextProtocolGuid,
                      NULL,
                      (VOID**)&Device2TextProtocol);


   if (EFI_ERROR(Status)){
         Print(L"located DevicePathToTextProtocol fail\n");
         return Status;
  }
   //第二步，找出所有支持DiskIo的设备
 Status = gBS->LocateHandleBuffer(ByProtocol,
                             &gEfiDiskIoProtocolGuid,
                             NULL,
                             &NumHandles,
                             &ControllerHandle);
  if (EFI_ERROR(Status)){
         Print(L"No Disk\n");
         return Status;
  }
  第三步，遍历每个DiskIo设备，并打开设备上的DevicePathprotocol
  for(HandleIndex=0;HandleIndex<NumHandles;HandleIndex++){

         Status = gBS->OpenProtocol(ControllerHandle[HandleIndex],
                                  &gEfiDevicePathProtocolGuid,
                                  (VOID**)&DiskDevicePath,
                                  gImageHandle,
                                  NULL,
                                  EFI_OPEN_PROTOCOL_GET_PROTOCOL);
          if (EFI_ERROR(Status)){
                  continue;
                }
            {
              TextDevicePath = Device2TextProtocol->
                   ConvertDevicePathToText(DiskDevicePath,TRUE,TRUE);
                Print(L"%s\n",TextDevicePath);
                if(TextDevicePath)
                {
                    gBS->FreePool(TextDevicePath);
                }
            //遍历设备路径DiskDevicePath里的各个设备节点
              }
                WalkthroughDevicePath(DiskDevicePath,PrintNode);
                Print(L"\n\n");
```

```
156                 }157
158   return Status;
159
160 }
```

收起 ^

The output shows:

```
FS0:\> DevicePathTool.efi
################################################
################################################
PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x0)/Pci(0x3,0x0)/Pci(0x0,0x0)/USB(0x0,0x0)
(2 1)/(1 1)/(1 1)/(1 1)/(1 1)/(3 5)/

PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x0)/Pci(0x3,0x0)/Pci(0x0,0x0)/USB(0x0,0x0)/HD(1,GPT,24B3671E-9516-41CF-839D-14840BBDE7D4,0x8
0
0,0x1D0CFDF)
(2 1)/(1 1)/(1 1)/(1 1)/(1 1)/(3 5)/(4 1)/

PciRoot(0x0)/Pci(0x2,0x0)/Pci(0x0,0x0)/Sata(0x0,0xFFFF,0x0)
(2 1)/(1 1)/(1 1)/(3 18)/

PciRoot(0x0)/Pci(0x2,0x0)/Pci(0x0,0x0)/Sata(0x0,0xFFFF,0x0)/HD(1,GPT,7D46FEBF-6A54-4DD9-8812-193EFD5C37F9,0x800,0xF42000)
(2 1)/(1 1)/(1 1)/(3 18)/(4 1)/

PciRoot(0x0)/Pci(0x2,0x0)/Pci(0x0,0x0)/Sata(0x0,0xFFFF,0x0)/HD(2,GPT,F8D3B4B3-1906-43C5-9864-C31FBC49AA17,0xF42800,0x1E8800)
(2 1)/(1 1)/(1 1)/(3 18)/(4 1)/

PciRoot(0x0)/Pci(0x2,0x0)/Pci(0x0,0x0)/Sata(0x0,0xFFFF,0x0)/HD(3,GPT,AAF03549-6191-4596-99C1-F314DD099D00,0x112B000,0xDBCB000)
(2 1)/(1 1)/(1 1)/(3 18)/(4 1)/

PciRoot(0x0)/Pci(0x2,0x0)/Pci(0x0,0x0)/Sata(0x0,0xFFFF,0x0)/HD(4,GPT,89AA1729-2305-45DF-A31F-53A62EB6C057,0xECF6000,0x186000)
(2 1)/(1 1)/(1 1)/(3 18)/(4 1)/
```