UEFI Principles and Programming Practice--File Operation

原创 Anthony Posted on 2021-11-29 13:45:14 Read 2.9k Collection 7 Likes

Category Column: UEFI Article Tags: UEFI

摘要 This article introduces how to use EFI_SIMPLE_FILE_SYSTEM_PROTOCOL to operate the FAT file system in the UEFI environment, including the basic steps of how to obtain the root directory handle, create and read files. It focuses on showing the practical application examples of the Write and Read functions.

File operations mainly focus on some key points

UEFI has built-in EFI_SIMPLE_FILE_SYSTEM_PROTOCOL for operating FAT file system. Through OpenVolume in this protocol, we can get the root directory handle on the FAT file system. The directory handle (FEI_FILE_PROTOCOL) contains the file operation interface for operating files in the directory.

Use EFI_SIMPLE_FILE_SYSTEM_PROTOCOL


```
EFI_STATUS Status;
EFI_SIMPLE_FILE_SYSTEM_PROTOCOL   *SimpleFileSystem;
EFI_FILE_PROTOCOL *Root;


Status = gBS->LocateProtocol (
                  ByProtocol,
                  &gEfiSimpleFileSystemProtocolGuid,
                  NULL,
                  &SimpleFileSystem
                  );
if (EFI_ERROR(Status)){
  return Status;
}

Status = SimpleFileSystem->OpenVolume(SimpleFileSystem,&Root);
```


After getting the pointer to the EFI_FILE_PROTOCOL instance of the root directory on the partition file system, you can operate the files on the partition

Write a file using the Write function


```
EFI_STATUS Status;

UINTN BufSize;

CHAR16 *Textbuf = (CHAR16*)L"this is test file\n";

EFI_FILE_PROTOCOL *FileHandle =0;

//创建新文件，如果文件已经存在，则打开

Status = Root->Open(Root,
                        &FileHandle,
                        (CHAR16)L"testfile.txt",
                        EFI_FILE_MODE_CREATE | EFI_FILE_MODE_READ | EFI_FILE_MODE_WRITE,
                         0);

if(FileHandle && !EFI_ERROR(Status)){
            BufSize = StrLen(Textbuf) *2;

Status = FileHandle->Write(
                FileHandle,&BufSize,Textbuf);

Status = FileHandle->Close (FileHandle);
}
```

Reading files with the Read function


```
EFI_STATUS                    Status;
  EFI_SIMPLE_FILE_SYSTEM_PROTOCOL *ptSFS;
  EFI_FILE_PROTOCOL               *ptRootFile;
  EFI_FILE_PROTOCOL               *ptFile;
  EFI_FILE_INFO                   *FileInfo;
  UINTN                           FileInfoSize;
  UINT8                           *Buffer;

 Status     = EFI_SUCCESS;
  ptRootFile = NULL;
  ptFile     = NULL;
  FileInfo   = NULL;
  HandleBuffer = NULL;

 Status = ptSFS->OpenVolume(ptSFS, &ptRootFile);
      if (!EFI_ERROR (Status)) {
        Status = ptRootFile->Open(
                   ptRootFile,
                   &ptFile,
                   FileName,//你的文件名，带目录
                   EFI_FILE_MODE_WRITE|EFI_FILE_MODE_READ,
                   0
                   );

 if (!EFI_ERROR (Status)) {
    FileInfo = NULL;
    FileInfoSize = 0;
    Status = ptFile->GetInfo (
                       ptFile,
                       &gEfiFileInfoGuid,
                       &FileInfoSize,
                       FileInfo
                       );
    if(Status == EFI_BUFFER_TOO_SMALL){
      FileInfo = AllocateZeroPool(FileInfoSize);
      if(FileInfo == NULL){
        Status = EFI_OUT_OF_RESOURCES;
      } else {
        Status = ptFile->GetInfo(
                           ptFile,
                           &gEfiFileInfoGuid,
                           &FileInfoSize,
                           FileInfo
                           );
      }
    }
  }

  Status = ptFile->Read(ptFile, &FileInfo->FileSize, Buffer);

ptFile->Close(ptFile);
ptRootFile->Close(ptRootFile);
```


The use of this read function is different from that in the book. This BIOS practical application will be discussed later.