

基于 IRP 的 Windows 设备驱动程序文件操作的实现

王兰英, 居锦武

(四川理工学院计算机学院, 四川 自贡 643000)

摘要: 文件是数据存储的基础, 处于 Windows 操作系统内核中的设备驱动程序, 对文件系统的访问不像应用程序那样方便。文章介绍了 Windows 设备驱动程序通过建立自定义 I/O 请求包 (IRP), 并将 IRP 发送到文件系统驱动程序的方式, 实现对文件的内核级操作的方法, 该方法可实现文件建立、读、写、删除和改名等功能。文章给出了基于微软设备驱动程序开发工具实现的 IRP 包的建立、发送和完成的源码。

关键词: 驱动程序开发工具; 驱动程序; 内核; 文件; I/O 请求包

中图分类号: TP311

文献标识码: A

引言

文件是操作系统的核心, 也是数据存储的基础, 文件基于磁盘或是半导体闪存存储, 具有非易失性, 不会因掉电等原因而导致数据丢失, 涉及到硬件的程序经常要将采集的数据或是日志数据存入文件。微软公司的 Windows 操作系统工作于 CPU 的保护模式下, 操作系统内核及设备驱动程序工作于 0 环, 称为核心态; 应用程序、动态链接库工作于 3 环, 称为用户态。操作系统通过环级保护来实现对用户态程序的访问权限限制。工作于核心态的设备驱动程序^[1]是操作系统的信任部件, 其访问权限不受任何限制。Windows 操作系统提供了多种多样的 API 供应用程序实现对文件的操作, 如创建、删除、改名、读和写等。但这些 API 工作于用户态, 只能供应用程序使用, 设备驱动程序无法使用。文章首先简要介绍 Windows 操作系统的基本结构, 然后介绍了在设备驱动程序中通过建立自定义的 IRP, 并将它发送到文件系统驱动程序, 以实现核心态文件操作的方法。

紧密的内核组件是 I/O 管理器, I/O 管理器支持分层驱动程序模型, I/O 管理器针对每个 I/O 请求建立一个 I/O 请求包 (IRP), 通过把 IRP 发送给驱动程序栈的顶层, 并由上向下传递, 最后由分层驱动程序中的某一层来完成 I/O 请求。

1 Windows 系统基本结构

Windows 操作系统采用结构化、模块化、层次化的设计思想。Windows 操作系统的基本结构如图 1 所示, 从层次上来看, 整个操作系统被分为多个层次, 分别是用户层, 内核层, 硬件抽象层。与设备驱动程序关系最

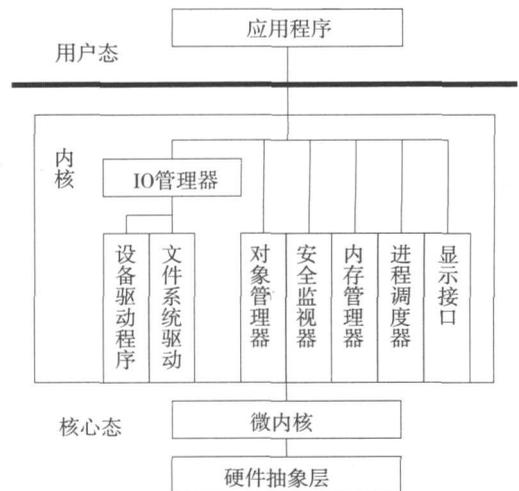


图 1 Windows 操作系统的结构

设备驱动程序与文件系统驱动程序都接受来自 I/O 管理器发来的 IRP, 对 IRP 的处理是驱动程序的主要工作。处于用户态的应用程序在调用文件操作 API 时, 将操作请求下载到处于内核态^[2]的 I/O 管理器, I/O 管

理器根据来自用户态的请求,建立相应的 IRP 包,并将 IRP 发送到文件系统驱动程序处理。文件系统驱动程序分析 IRP 的内容来了解所要执行的操作的具体情况,并完成相应的处理。文件系统驱动程序本身是一个分层的驱动模型,上层是文件系统过滤驱动程序,中间是 NTFS、FAT 等文件系统驱动程序,底层是存储设备驱动程序^[3]。这些驱动程序共同协作完成用户的请求。

文件系统驱动程序与其它的设备驱动程序一样,并不关心 IRP 的来源。因此,设备驱动程序可以通过创建自定义的 IRP,并将其发送给文件系统驱动程序,文件系统驱动程序一样会识别该 IRP,并完成它。这样也就完成了设备驱动程序自己的文件操作请求。

2 I/O 请求包 (IRP) 的建立、发送、完成

I/O 请求包是 I/O 系统用来存储处理 I/O 请求所需信息的数据结构^[4]。当请求者调用 I/O 服务时, I/O 管理器就构造一个 IRP 来表示该请求在整个系统中要进行的操作。IRP 是从非分页内存池^[5]中分配的可变大小的结构,包括两个部分:一个头部区域,包括一般的管理信息;一个或多个参数区域,称为栈单元。头部区域中的 AssociatedIrp、SystemBuffer、MdlAddress 和 UserBuffer 字段用于管理驱动程序对数据缓冲区的使用方式。栈单元中的主功能代码 MajorFunction 参数表明了 IRP 所要执行的操作。

2.1 IRP 的建立

设备驱动程序自己构造 IRP 可以采用多种方法^[6],以下三个函数都可以构造一个 IRP:

```
IoBuildAsynchronousFsRequest
IoBuildDeviceIoControlRequest
IoBuildSynchronousFsRequest
```

这些函数将根据参数表的情况来自动设置所构造的 IRP 的头部区域和栈单元的内容,可以方便 IRP 的建立工作,但这些函数只能产生一些特定功能的 IRP。在一些特殊情况下,需要从非分页内存中直接分配 IRP,并手动填充它的各个参数域。用函数 IoAllocateIrp 构造 IRP 的实例代码如下:

```
irp = IoAllocateIrp ( fileobject->DeviceObject->StackSize); // 从非分页内存分配 IRP
if ( fileobject->DeviceObject->Flags & DO_BUFFERED_IO) // 根据 I/O 操作所使用的缓冲方式来设置相应的标志
{ irp->AssociatedIrp.SystemBuffer = buffer } // 缓冲方式
else if ( fileobject->DeviceObject->Flags & DO_DIRECT_IO)
{ mdl = IoAllocateMdl ( buffer, count, 0, 0);
```

```
MmBuildMdlForNonPagedPool (mdl);
irp->MdlAddress = mdl } // 直接 IO 方式,不用缓冲
```

```
else { irp->UserBuffer = buffer }
// 不使用直接 IO 方式,也不用缓冲,直接传递地址
irp->FileObject = fileobject // 填充 IRP 内各个参数域
```

```
irp->MajorFunction = IRP_MJ_READ;
irp->MinorFunction = IRP_MN_NORMAL; // 0
irp->Parameters.Read.ByteOffset = offsetused;
irp->Parameters.Read.Key = 0;
irp->Parameters.Read.Length = count;
```

2.2 IRP 的发送和完成

将 IRP 发送给文件系统驱动程序可调用函数 IoCallDriver,但首先要获得所要操作的文件的文件对象 FileObject 同时得到文件所属文件系统^[7]的设备对象 DeviceObject,然后再调用 IoSetCompletionRoutine 函数注册一个 I/O 完成例程^[8],当 IRP 被文件系统驱动程序完成时,该 I/O 完成例程将得到执行,从而使设备驱动程序有机会了解 IRP 的完成情况,并做相应的处理。相应代码如下:

```
// 首先创建文件,并得到文件的句柄 filehandle
IoCreateFile (&filehandle, access & obj & ios, 0, FILE_ATTRIBUTE_NORMAL, share, FILE_OVERWRITE_IF, 0, 0, 0, 0, 0);
// 根据文件句柄 filehandle 得到文件对象 fileobject
ObReferenceObjectByHandle (filehandle, GENERIC_READ, *IoFileObjectCcbType, KernelMode (PVOID*) &fileobject);
IoSetCompletionRoutine (irp, IoCompletion, &event, 1, 1, 1); // 设置完成例程
IoCallDriver (fileobject->DeviceObject, irp); // 将 IRP 发送到文件所属的文件系统驱动程序。
```

3 文件操作程序

在 IRP 建立时,通过给栈单元中的主功能代码 MajorFunction 参数赋不同的值,即可指定文件系统驱动程序对文件完成相应的操作。其值与所完成功能的对应见表 1。

表 1 MapIRFunction 参数的功能

MajorFunction 参数	所实现的功能
IRP_MJ_READ	读文件内容
IRP_MJ_WRITE	改写文件内容
IRP_MJ_CREATE	建立文件
IRP_MJ_SET_INFORMATION	删除
IRP_MJ_SET_INFORMATION	改名

当将 IRP 的主功能代码 MajorFunction 赋值为 IRP_MJ_SET_INFORMATION, 并将 IRP 中的 Parameters SetFile FileInformationClass 字段设置为 FileRenameInformation, IRP 的功能是将文件改名。如果将主功能代码 MajorFunction 参数赋值为 IRP_MJ_SET_INFORMATION, 并将 IRP 中的 IrpSp->Parameters SetFile FileInformationClass 字段设置为 FileDispositionInformation, IRP 的功能是删除文件。以下为实现文件改名操作的代码, 其它操作的代码与此类似。

```
irp->MajorFunction= IRP_MJ_SET_INFORMATION; //设置 IRP 的主功能代码 MajorFunction 及其它参数
```

```
irp->Parameters SetFile Length =
sizeof(FILE_FILE_RENAME_INFORMATION);
irp->Parameters SetFile FileInformationClass =
FileRenameInformation;
```

4 结束语

文章分析了 Windows 的体系结构, 分析了 I/O 管理器对 IRP 的管理方式, 介绍了驱动程序对 IRP 的处理过程。在 Windows 操作系统下, 设备驱动程序通过自建 IRP, 并将其发送到文件系统驱动程序, 来实现内核级文件操作的方式进行了详细分析。文章给出了具体的 IRP 的构造、发送和完成方法, 同时还给出了据此实现

文件操作的驱动程序源码。文章使用的方法具有通用性和易用性, 并在 VC++ 开发的工业控制系统中得到应用, 取得了良好的效果。本文介绍的方法对于内核驱动程序实现文件操作的开发具有实际的参考价值。

参考文献:

- [1] 蹇红梅, 居锦武, 王兰英. Windows 设备驱动程序的开发[J]. 四川理工学院学报: 自然科学版, 2007, 20(4): 11-13
- [2] 王兰英, 居锦武. Windows 内核模式驱动程序运行环境的分析[J]. 微计算机信息, 2005, 21(11): 201-202, 197
- [3] 王德明, 刘淳, 邱俊山. 基于过滤器驱动程序的文件保护系统软件设计[J]. 四川理工学院学报: 自然科学版, 2006, 19(4): 78-80
- [4] 文臣. 基于 Windows 2000 操作系统的 PCI 卡 WDM 驱动程序开发[J]. 四川理工学院学报: 自然科学版, 2005, 18(1): 52-56
- [5] 郭益昆. VC++. NET 开发驱动程序详解[M]. 北京: 希望电子出版社, 2002
- [6] [美] Microsoft Corporation Windows 2000 驱动程序开发大全[M]. 北京: 机械工业出版社, 2001
- [7] [美] Solomon D A. Windows 2000 内部揭秘[M]. 北京: 机械工业出版社, 2001
- [8] [美] Cant C. Writing Windows WDM Device Drivers[M]. 北京: 机械工业出版社, 2000

Realization on Operation for Windows Device Drivers Based on IRP

WANG Lan-ying, JU Jin-wu

(School of Computer Science, Sichuan University of Science & Engineering, Zigong 643000, China)

Abstract Files are the basis of datamemorizing, the device drivers in the windows accessing system are not as convenient as visiting the document systems. To realize the operation on the extreme kernel of the document, this paper introduces how to deve the device, send IRP to document system drivers through I/O package, which includes the functions of setting up, reading, writing, deleting and renaming the files. And it also works out the source codes of IRP package's establishment, sending and completing, which is based on development of the microsoft device drivers.

Key words DDK; driver; kernel; file; IRP