# 关于KAKU Installer的内部参考

这是一个自解压文件生成器，即它本身是一个.exe文件，但是被附加了大量的额外数据，这部分数据被称之为附加数据段（Additional Data Segment，简称ADS）。

本项目分为两个工程，其一为自解压程序本身，其二为将数据挂载至自解压程序中的挂载程序，本质上，前者为解压缩程序，后者为压缩程序。

本文论述了自解压程序的运行原理与设计思路。

# 第一章 概要设计

该程序的功能就是将自身携带的数据按照规定的数据结构释放至磁盘中，如果数据经过加密、压缩则进行解密、解压。

KAKA Installer自解压程序是被挂载程序生成的，由程序段+数据段组成的可运行程序，其中程序段负责将数据段的数据根据后者的记录与用户的要求释放至磁盘的规定位置。程序段基于KAKU Framework编写。

KAKA Installer自解压程序被设计为Windows操作系统上运行的x64程序。

### 1.1系统架构设计

系统共包括两个部分，分别为IO部（Input & Output Part，简写为IOP）与数据操作部（Data Operator Part，简写为DOP）。其中IO部负责数据的流动，即将ADS数据读入内存并将经过数据操作部处理的数据根据其要求输出至磁盘，在整个系统中仅有IO部有权对ADS数据进行磁盘操作。数据操作部负责对ADS数据进行处理，包括解压缩、建立文件组织、校验等。

### 1.2 系统功能模块概要设计

本章节对系统的两个部分，即IO部与DOP部进行讲解，主要分析二者的运行逻辑并。其中IO部作为内存与磁盘间的唯一中介进行数据交流，包含in模块与out模块。DOP部负责解压功能与建立文件结构功能，对应的模块为：解压模块与建立文件结构模块。

在章节最后提供了系统类图，并简述了系统中各类之间的关系。

### 1.2.1 IO部（Input & Output Part）

IO部的任务非常简单，作为内存与磁盘间的中介进行数据交流。具体来说，IO部中的in模块从磁盘中找到ADS的位置并将数据读入内存中并记录地址，随后通知DOP对该地址的数据进行处理。在DOP完成对数据的处理后会将新的数据放入内存中，再通知IO部将这些数据重新释放至磁盘中，之后，IO部的out模块便会将DOP提供的数据根据其数据结构与其它记录释放至内存中，其工作流程如图1、2所示。

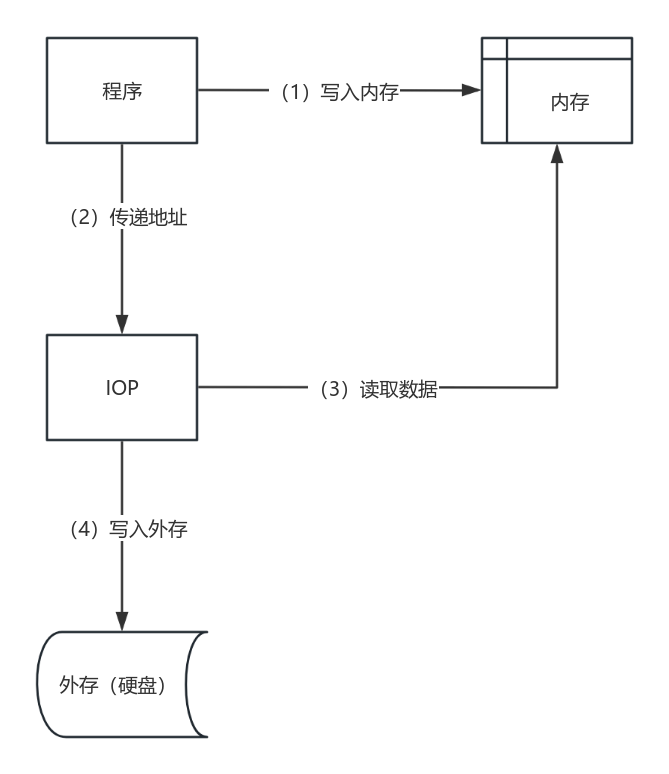


图 1 IOP输出流程

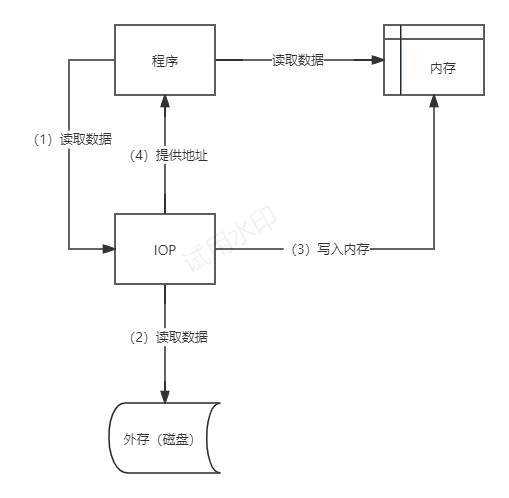


图 2 IOP输出流程

基于以上业务逻辑，IOP的设计被确定为：对磁盘IO功能进行针对业务需求的简单封装，增加其与业务本身的耦合性。

### 1.2.2 DOP部（Data Operator Part）

DOP部负责两大业务：解压功能与建立文件结构功能，负责实现二者功能的模块为：解压模块与建立文件结构模块。

解压模块的任务是根据数据采用的压缩算法对数据进行解压缩，获取原数据。一般来说，数据采用zip或7z压缩算法。

建立文件结构模块（Creating Directory Structure Module，简写为CDSM）负责将经过解压处理的数据根据其中记录的逻辑结构数据重新组织为新的逻辑数据结构，例如，原数据为：

“第1组数据：abc；下一组数据：5”

“第2组数据：ghi；下一组数据：4”

“第3组数据：mno；下一组数据：无”

“第4组数据：jkl；下一组数据：3”

“第5组数据：def；下一组数据：2”

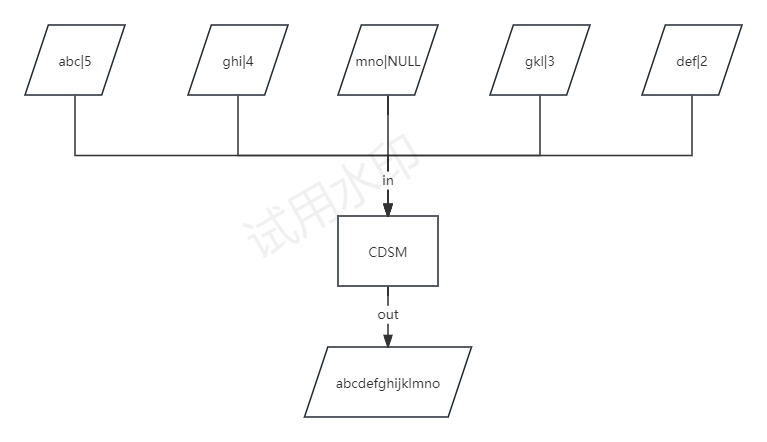


图 3 CDSM实例

如图3所示，CDSM会将其整理为“abcdefghijklmno”。在实践中的情况当然要复杂的多，因为ADS数据的逻辑结构是非线性的（一般来说是树状结构，以对应操作系统中的文件结构）。

### 1.2.3 系统类图

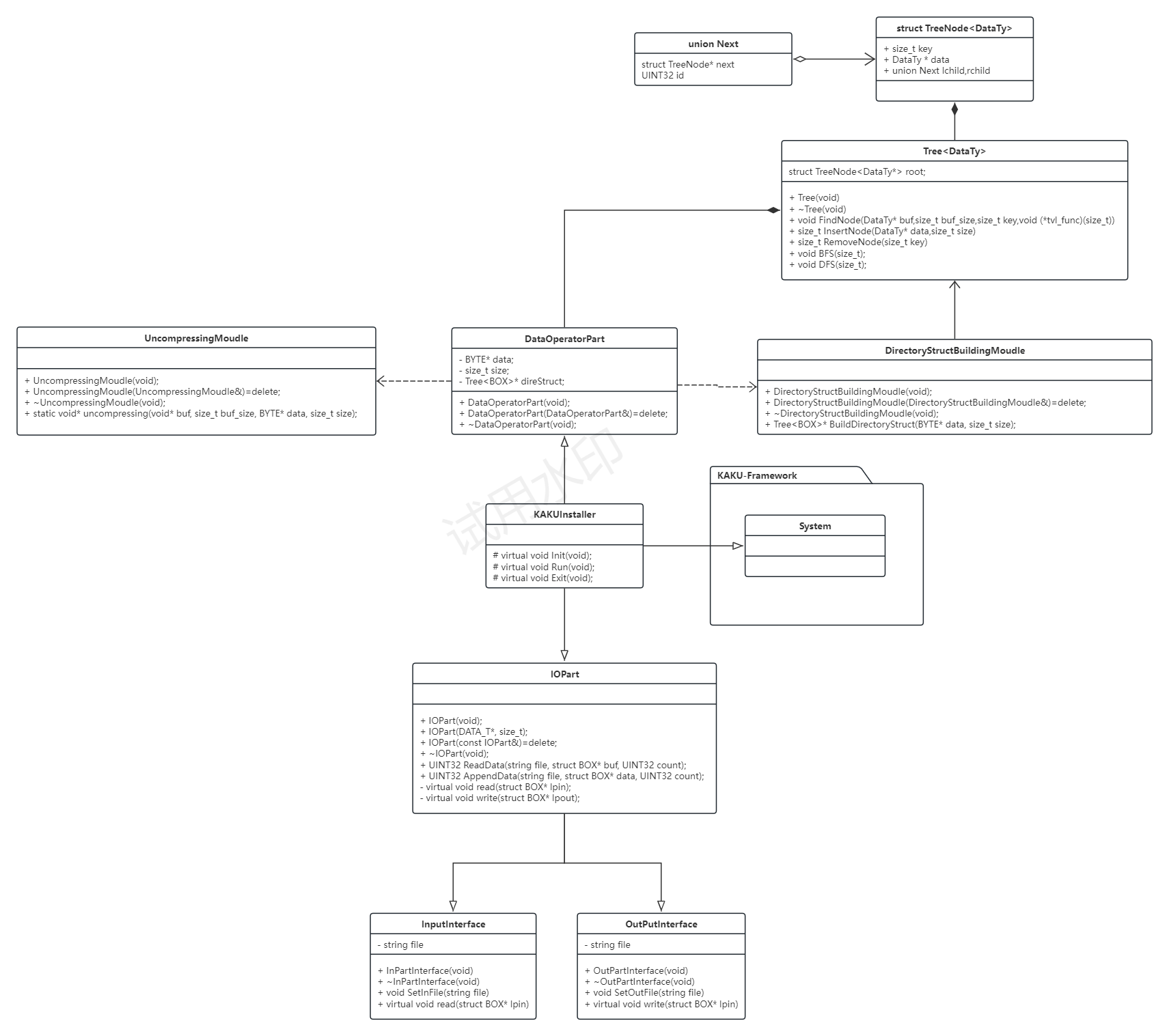


图 4 系统类图

KakuInstaller系统类图如图4所示，KAKUInstaller类继承DOP类、IOP类与System类。其中System类为KAKU Framwork的对外接口，与其相关情况可以查看文献[[[1]](#endnote-0)],在此不做赘述。

DOP类与UncompressingMoudle类、DirectoryStructBulidingMoudle类存在依赖关系。具体的说，UncompressingMoudle类提供一个静态方法uncompressing用于对数据进行解压缩；DirectoryStructBulidingMoudle类提供一个静态方法BuildDirectoryStruct用于构建文件结构。Tree类与TreeNode结构体之间存在聚合关系，二者被设计为用于维护二叉树的数据结构，DOP类与Tree类存在聚合关系，DirectoryStructBulidingMoudle类与Tree类存在依赖关系。

IOP类实现了两个接口，分别为InputInterface接口和OutPutInterface接口。这两个接口分别包含一个私有成员file与对应的set方法。InputInterface接口包含一个公有的read虚方法，OutPutInterface接口包含一个公有的write虚方法，二者在IOP类中被实现。InputInterface接口和OutPutInterface接口均依赖由库提供的fstream类。

在图4之外，还有一系列用于记录数据的数据结构，如BOX类等，它们与图4中的大多数类普遍存在依赖关系，其具体结构可以参看本文1.2.1章节。

# 第二章 详细设计

本章主要介绍了KAKU Installer的详细设计，包括工作流程，内外存数据结构，核心类工作原理等。

## 2.1 工作流程

本程序的基本工作流程如图5所示，程序运行的时候首先会获取自身的数据，即将进程映像作为输入，然后寻找ADS段并获取其中的数据，对其中的数据进行处理后再重新释放至磁盘中。

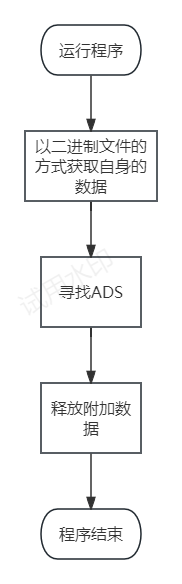


图 5 KAKU Installer总流程图

在实现过程中的主要难点出现在第（3）步和第（4）步，即寻找ADS和释放附加数据。

寻找ADS步骤的难点在于用何种方式识别ADS，目前的技术路线有两条：其一为使用标记法，即在ADS前添加一段标志数据（魔数），可以为0xFDFDFDFD等。其二为固定相对地址法，即固定文件的某一部分之后为ADS。

对于释放附加数据步骤，其难点在于ADS的数据结构的设计。在设计中，被压缩数据是一系列如图6所示的以树形结构组织的路径和文件，其结构被记录在ADS的第一个块中。

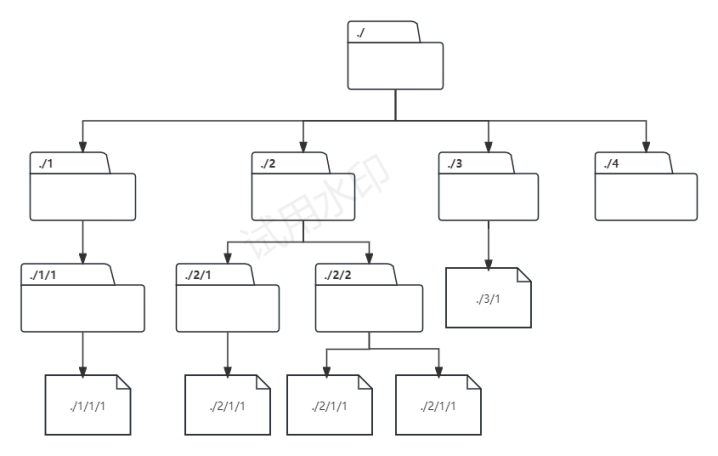


图 6 一个典型的文件路径结构

#### 1.2.1 基本数据结构单元

组成ADS的基本数据结构单元为“块（BOX）”。每个块包含两部分：头域（HEAD）和数据域（DATA）。每个块的数据部分尺寸是不固定的，由头中的一个字段记录。

其中HEAD占32Byte，其中前4Byte中的记录为BOX的id，该数据为主键，由于使用unsigned int处理该部分数据，故本程序最大支持4294967295个块。id是BOX中最重要的数据之一，程序将根据id组织ADS的逻辑结构（树状结构）。

第5-8Byte是为了满足后续需要而预留的，默认使用0xEE填充。

第9-16Byte为DataSize，其记录了DATA域的尺寸，格式为size\_t（unsigned long long）。程序将根据该数值读取DATA域，该部分的数据异常将导致后续数据读取的错位。

第17-24Byte为DATA域的数据校验码。

第25-32Byte为使用0xFF填充的结束标志，若读取BOX时发现该部分数据异常说明发生了数据错位错误。

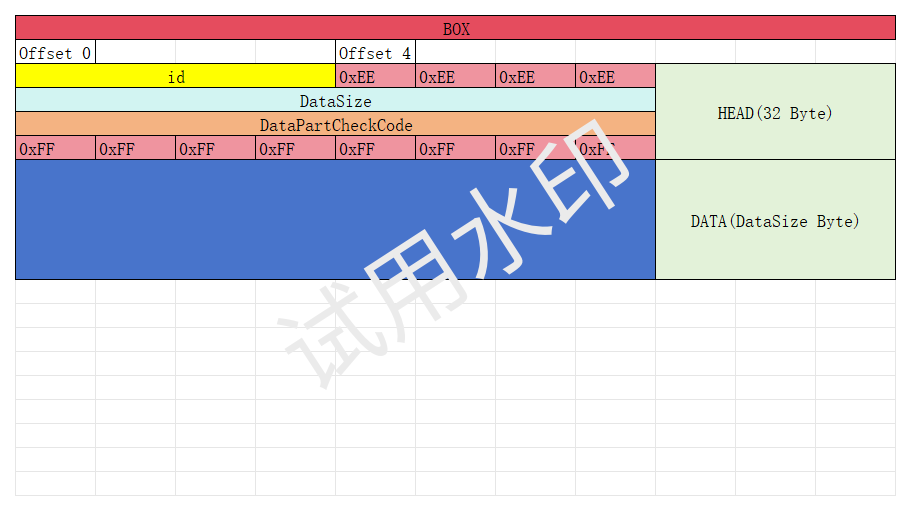


图 7 BOX数据结构

BOX的类图如图8所示：

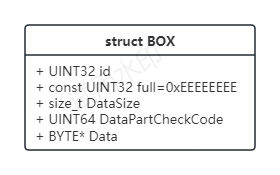


图 8 BOX类图

目录块是一个特殊的块，其中记录了ADS的逻辑结构。他是ADS的第一个块，id为0。其DATA域中记录了一组树状链表DirTree，其中：

第1-8字节为当前节点的主键。

第9-16字节记录了兄弟节点的主键。

第17-24字节记录了子节点的主键。

对于树状链表的节点数据结构为二叉链表，采用“兄弟-孩子记录法”，即节点中记录的是左孩子节点和兄弟节点的主键[[[2]](#endnote-1)]。

在磁盘中，id，next，deep均采用IdType类型记录对应节点的id，当DirTree为加载至内存中时，id将被替换为数据在内存中的虚拟地址。

各个节点在DATA域内部采用线性顺序存储，即采用线性表作为数据结构。

考虑到DATA域的最大尺寸为4294967295Byte，其最多支持178956970个节点。计算公式如下：

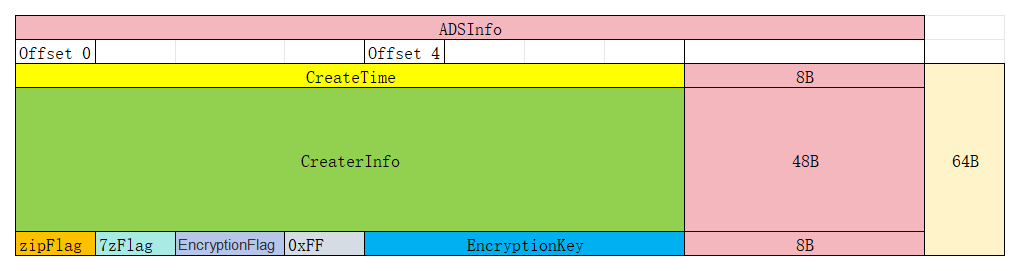


图 10 ADSInfo

#### 1.2.2 ADS读取流程

由于整个ADS具有如下结构，读取的过程本质上就是按顺序读入结构中的各个部分的过程。

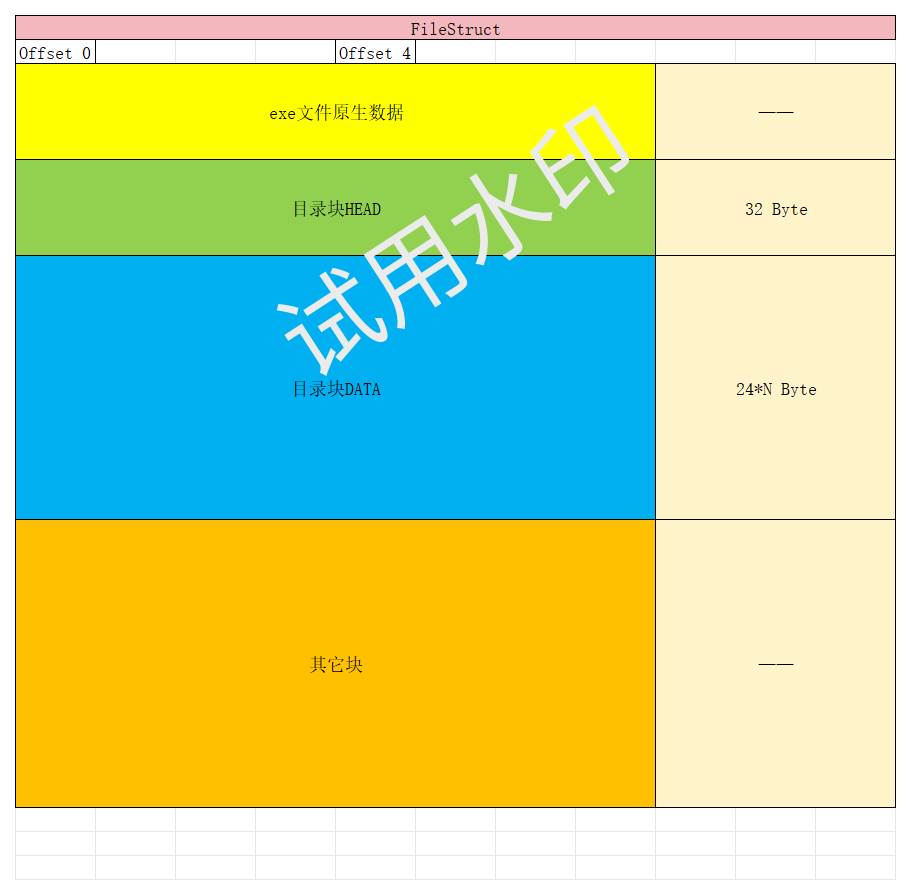


图 11 FileStruct

如果数据是被压缩过的则现进行解压缩。

如果数据有加密则在此时进行解密。

首先读取目录块，并根据DATA域中的情况构建树形数据结构DirTree，并在后续的操作中将从其它块中读取数据并装载到DirTree中。

在读取完毕后，按照DirTree将数据（路径和文件）释放至磁盘中即可。

# 参考文献

1. [] 郭蕴泽.自主开发的具备并发能力的系统框架——以阅读器的实现为例[D].黑龙江大学,2024. [↑](#endnote-ref-0)
2. [] 严蔚敏,李冬梅,吴伟民.数据结构[M].北京:人民邮电版社,2015:134-134. [↑](#endnote-ref-1)