文章编号:1004-4116(2021)04-0085-05

基于 MD5 码地质资料(电子文档)的完整性校验

姚兴华,张晓娟,刘彩英
(甘肃省地矿局第三地勘院,甘肃 兰州 730050)

摘 要:对地质资料的电子文档,文件长时间保存和复制过程中可能发生的错误,进行核对校验,保证电子文档的完整性和准确性,采用Python 3. 6. 8开发,在地质资料电子文档归档初期做好校验文件,之后可以用于该资料的校验,提高地质资料电子文档的管理质量。

关键词:MD5码;Python
中图分类号:P 621 **文献标志码**:B

地质资料是指在地质、矿产、物化探、测绘、水工环等工作中形成的文字、数据、图表、影像等形式的原始地质资料、成果地质资料和岩矿芯、各类标本等实物地质资料。地质资料是地质勘查的第一手成果,真实的记录和反映了地质工作的过程。随着现代信息技术在地质领域的广泛应用,形成了大量的地质资料电子文档。

地质资料电子文档分为文档和数据库两类,其中文档类资料占主要类型,文档类资料又分文字类和图件类,文字类主要有 Word、Excel、Pdf、Jpg 等格式,图件类以 MapGis、AutoCAD、ArcGis 等软件的文件为主。皆以树型目录形式保存在存储介质中,由于地质资料的复杂性和异构性会形成大量计算机文件。这些文件,在长期保存或更换存储介质时,会有一定概率的出错,或可能在阅览时被不经意的修改或者破坏,这就需要文件校验来保证电子文档的完整性和准确性,我们采用 Python 语言开发了一款文档类校验软件,采用记录文件保存路径和文件的 MD5 码来核对、校验电子文档资料。

MD5 是 message-digest algorithm 5(信息—摘要算法)的缩写,被广泛用于加密和解密技术上,它可以说是文件的"数字指纹"。无论是可执行程序、图像文件、临时文件或者其他任何类型的文件,都有且只有一个独一无二的 MD5 信息值,并且如果这个文件被修改过,它的 MD5 值也将随之改变。因此,我们可以通过对比同一文件的 MD5 值,来校验这个文件是否被"篡改"过。

Python 是 20 世纪 90 年代初由荷兰的吉多·范罗苏姆(Guido van Rossum)创建的一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。是 FLOSS(自由/开放源码软件)之一,可以自由地发布这个软件的拷贝、阅读它的源代码、不牵扯版权。

1 技术介绍

现有的采用 MD5 码校验文件的软件是比较多见的,如:RapidCRC 等,都只针对文件进行 MD5 校验,对于地质电子文档资料的文件夹嵌套和文件的数量较大特性来讲操作较为繁琐,且没有记录文件的保存路径,这对于如 MapGis 这类采用工程文件管理各类文件的软件,路径错误会导致信息的缺失。所以我们开发的校验软件即保存文档的存储路径也保存文件的 MD5 码,软件采用 Python 3. 6. 8 开发,图形界面采用 Python 自带 tkinter 库,文件的 MD5 码计算采用 hashlib 库。

2 设计和实现

文件的 MD5 码计算,代码如下:

收稿日期:2021-08-03
作者简介:姚兴华(1969 ~),男,学士,工程师,主要从事地质矿产勘查及信息化工作。E-mail:594236852@qq. com

```
    def file_md5(self, handle_file):
        """计算 handle_file 文件的 MD5 码"""
        hafiz = os. path. getsize(handle_file) // 1048576  # 获取文件的大小,转为 MB
        if hafiz > self. __boundary:  # 大于 31MB,分次读取
            m = hashlib. md5()
            with open(handle_file, 'rb') as fp:
                while True:
                    data = fp. read(self. __read_buff)  # 每次读 8MB
                    if not data:
                        break
                    m. update(data)  # 更新 MD5
        else:  # 小于 32MB 的文件,一次性读取
            with open(handle_file, 'rb') as fp:
                data = fp. read()
            m = hashlib. md5(data)  # 求文件的 MD5
        return m. hexdigest()
```

文件夹的遍历采用递归算法代码如下：

```
    def travel(self, cur_path):
        """遍历 self. work_path"""
        filer_list = os. listdir(cur_path)  # 返回文件夹包含的文件或文件夹的名字的列表
        for file in filer_list:  # 遍历列表
            abs_path = os. path. join(cur_path, file)  # 绝对路径
        if os. path. isdir(abs_path):  # 是文件夹继续遍历
            self. travel(abs_path)  # 递归调用
        else:
            ext_name = os. path. splitext(abs_path)[1][1:]. upper()  # 文件扩展名
            if ext_name not in self. __remove:  # 不属于排除的文件
                self. filecount + = 1 # 文件计数 + 1
                md5 = self. file_md5(abs_path)  # 文件的 MD5 码
                self. info. append([str(self. filecount), abs_path[self. __section:], md5])  # 存入列表
                self. writecheck()  # 写入 MD5 文件
```

按照校验文件 check. md5 核对文件夹的代码如下：

```
    def check_md5(self):
        """按照校验文件核对文件的 MD5 码"""
        chk_file = open(self. verify_file, 'r')  # 读取校验文件
        linens = chk_file. readlines()  # 全部读入
        if linens[1][5:]. strip() ! = self. work_path:
            TK. messagebox. showwarning(title = "提示", message = "资料文件夹和校验时文件夹不相同。")
        linens = linens[2:]  # 去掉前两行
        for line in linens:  # 第三行到最后一行
            self. filecount + = 1  # 文件数 + 1
            txt_line = line. strip(). split("|")  # 一行去空格,分解
```

```
            abs_path = os. path. join(self. work_path, txt_line[0])  # 绝对路径
            file_md5 = txt_line[1]  # 保存的 MD5
        if os. path. isfile(abs_path):
            md5 = self. file_md5(abs_path)  # 计算文件的 MD5
            if md5  = = file_md5:
                self. info. append([str(self. filecount), txt_line[0], "正确"])
            else:
                self. info. append([str(self. filecount), txt_line[0], "校验错误"])
            else:
                self. info. append([str(self. filecount), txt_line[0], "文件不存在"])
        chk_file. close()
```

校验文件的扩展名为. md5,采用文本形式保存于检验文件夹目录下,可以用记事本等文本编辑软件打开,摘要记录了生成该文件的日期、时间,选择目录为校验文件夹,之后的数据为文件的相对路径和校验码,以"|"分割,具体见图 1。

软件运行界面见图 2,MD5 码和校验按钮分别对应生成校验文件和以检验文件校验文件夹。

3　运行结果

电脑配置:i7 3630Q 2. 4G、8. 00GB 内存、普通笔记本机械硬盘,选择了一个 570MB,包含 266 个文件夹,2381 个文件(主要为 MAPGIS、Excel、Word、Jpeg 文件)的文件夹,计算 MD5 码生成校验文件用时大约是 6 秒,然后作者修改了其中一个 MapGis 图件的点文件(. WT),并删除了几个文件,运行软件,校验该文件夹用时约 5 秒, 精准的对之前的操作给出了校验错误和文件不存在的结论。见图 3。

4　结论

对于嵌套文件夹及多文件的校验做到准确计

图 1　校验文件的结构

Fig. 1　Structure of verification file

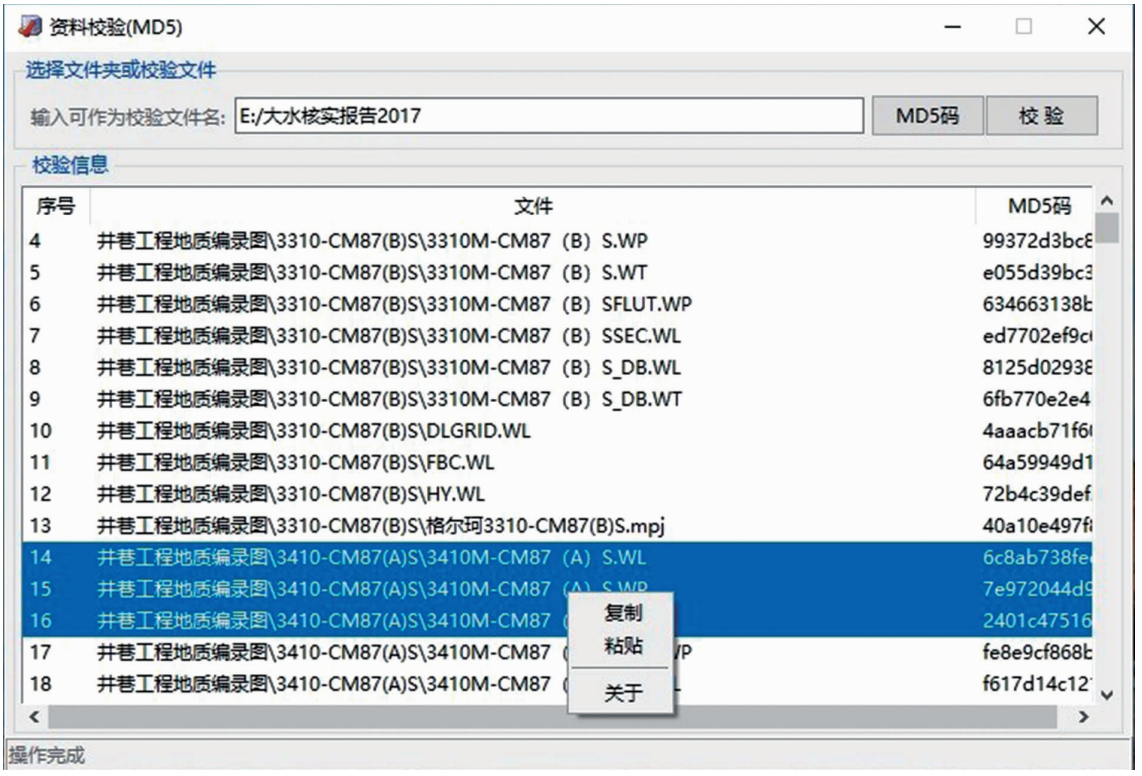

图 **2** 软件运行界面

Fig. 2 Interface of software operation

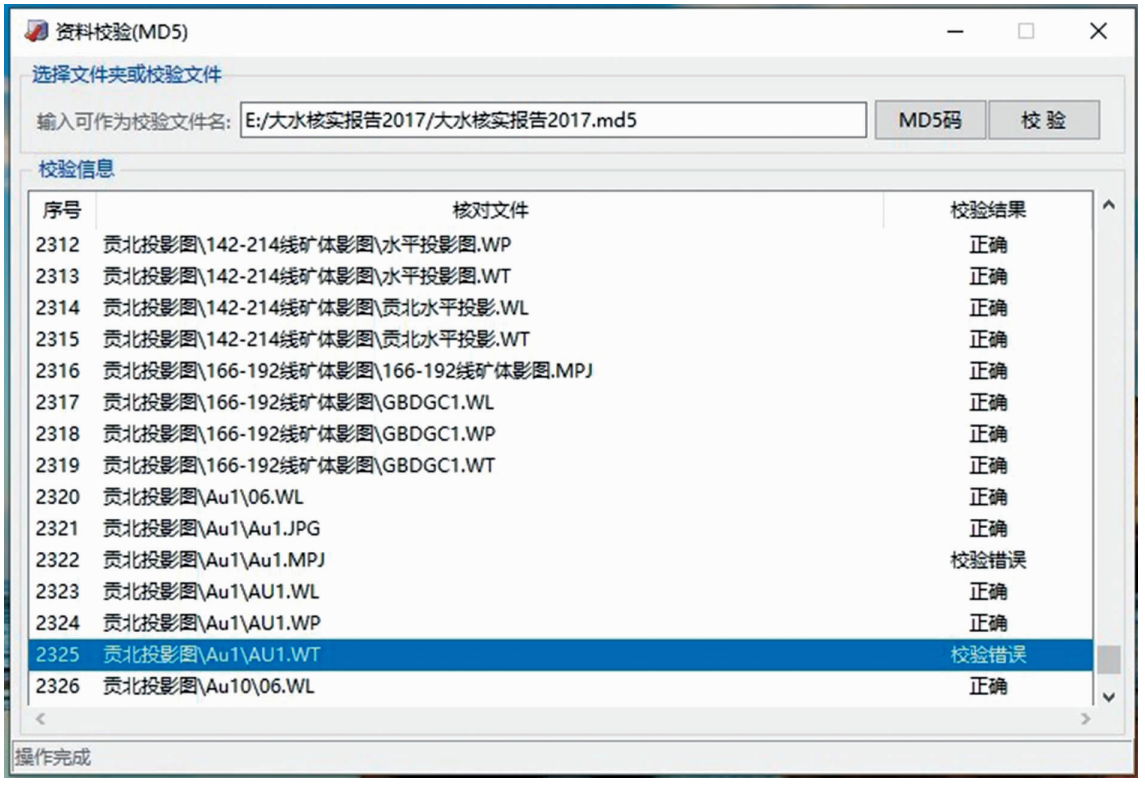

图 **3** 软件校验结果

Fig. 3 Verification results

算,对文件修改或删除只能提示,不能提供具体的恢复方案,这是需要进一步研究改进的地方。对 Python 图形界面 tkinter 库的运用,多个控件的相互绑定,剪切板的访问和赋值等有了更深入的了解,软件的运行效率和校验结论符合设计的预期,能满足文件夹嵌套和多文件的校验,具有一定的实用价值。

作者已将该软件采用 pyinstaller 封装为 exe 文件,可以脱离 Python 环境独立运行,有需要者可与作者联系,提供免费使用。

在软件的构思和开发过程中,作者得到了同事们的很多帮助,在此表示感谢。

参　考　文　献

[1] DA/T 41-2008,原始地质资料立卷归档规则[S]

[2] GB/T18894-2002,电子文件归档与管理规范[S]

[3] Magnus Lie Hetland [挪威]. Python 基础教程(第 3 版)[M]. 袁国忠 译. 北京:人民邮电出版社,2016

DATA INTEGRITY VERIFICATION OF GEOLOGICAL DATA (ELECTRONIC DOCUMENT) BASED ON MD5 CODE

YAO Xing-Hua, ZHANG Xiao-Juan, LIU Cai-Ying

(*The Third Institute of Geology and Mineral Exploration, Gansu Provincial Bureau of Geology and Mineral Exploration and Development, Lanzhou 730050, China*)

Abstract: Check and verify the errors that may occur in the process of long-term storage and copy of electronic documents of geological data, so as to ensure the integrity and accuracy of the electronic documents. Open source Python 3. 6 is adopted instead of the third-party library. In the initial stage of filing of electronic documents of geological data, the verification documents can be used for the verification of the data and promote the electronic documents of geological data.

Key words: MD5; Python