

# 基于递归与多线程的丢失文件查找设计

钟秀玉<sup>1</sup>, 陈月峰<sup>2</sup>

(1. 嘉应学院 计算机学院, 广东 梅州 514015;

2. 广东海洋大学 信息学院, 广东 湛江 524088)

**摘要:**在文件意外丢失的情况下,需在文件数量庞大的系统中查找丢失文件。根据操作系统对文件的管理以树型进行组织,即是一种递归的数据结构进行存储,递归的数据结构可使用递归的算法;同时,要在文件数量庞大的系统中进行查找,可用多线程技术以加快查找速度。在递归搜索过程中创建丢失文件链表及现存文件链表,开辟线程。当需要对丢失文件进行查找恢复时,可从丢失文件链表中快速找到丢失文件信息,参照现存文件链表对该文件进行查找恢复。实验结果表明,该方法具有一定的可行性和适用性。

**关键词:**递归;多线程;丢失文件;文件信息;查找

**中图分类号:**TP311

**文献标识码:**A

**文章编号:**1673-629X(2010)09-0098-04

## Search Design of Lost Files Based on Recursion and Multi-Thread

ZHONG Xiu-yu<sup>1</sup>, CHEN Yue-feng<sup>2</sup>

(1. Computer Institute, Jiaying University, Meizhou 514015, China;

2. Information College, Guangdong Ocean University, Zhanjiang 524088, China)

**Abstract:** In the file accident cases, searching the lost file is needed in a high number file of system. Use recursive method according to recursion data-structure of the file management organization form in the operating system and use multi-thread technology to locate quickly the lost file in many files. Creat the lost files linked list and existing files linked list in the course of a recursive search and set up thread. Finding the lost file information quickly when data recovery is needed, resume lost files refer to the existing file linked list. The method which this paper proposed is of certain feasibility and the serviceability after confirmed through the data simulation.

**Key words:** recursion; multi-thread; lost files; file information; search

## 0 引言

计算机普及率呈直线上升,误删除、误格式化、误分区等意外事故时常发生,造成数据破坏和丢失,带来巨大的经济损失。如何准确快速查找磁盘分区或目录中丢失的所有文件并对其进行恢复是人们非常关心的问题。根据文件存储结构特点,使用递归算法方便地进行文件搜索;同时,由于文件搜索涉及大量文件,为提高应用程序响应速度,使用多线程技术。

## 1 递归与多线程技术

### 1.1 递归

递归是一个过程或函数在其定义或说明中又直接

或间接调用自身的一种方法。它通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解,递归策略只需少量的程序就可描述出解过程所需要的多次重复计算,大大地减少了程序的代码量。一个递归算法必须有两个部分:递归出口和递归部分<sup>[1]</sup>。递归出口是一个明确的递归结束条件,只处理可以直接解决的问题;递归部分则包含对算法的一次或者多次递归调用,每一次的调用参数都在某种程度上更接近递归出口<sup>[2]</sup>。

### 1.2 多线程

多线程技术已经被 Windows 操作系统所支持。与进程相比,多线程技术是一种非常“节俭”的多任务操作方式。启动一个新进程必须分配给它独立的地址空间,建立众多的数据表来维护它的代码段、堆栈段和数据段,这是一种“昂贵”的多任务工作方式。而运行于一个进程中的多个线程,它们彼此之间使用相同的地址空间,共享大部分数据,启动一个线程所花费的空间远远小于启动一个进程所花费的空间,而且,线程间

收稿日期:2010-01-18;修回日期:2010-04-23

基金项目:广东省自然科学基金(9151009001000043);广东省科技计划项目(0911050400004)

作者简介:钟秀玉(1972-),女,副教授,硕士,CCF会员,研究方向为数据安全、数据库应用。

彼此切换所需的时间也远远小于进程间切换所需要的时间<sup>[3]</sup>。

线程之间共享数据空间。对不同进程来说,它们具有独立的数据空间,要进行数据的传递只能通过通信的方式进行,这种方式不仅费时,而且很不方便。线程则不然,由于同一进程下的线程之间共享数据空间,所以一个线程的数据可以直接为其它线程所用,这不仅快捷,而且方便。当然,数据的共享也带来其他一些问题,有的变量不能同时被两个线程所修改,这些正是编写线程程序时最需要注意的地方<sup>[4]</sup>。

多线程技术有效提高了应用程序响应。这对图形界面的程序尤其有意义,当一个操作耗时很长时,整个系统都会等待这个操作,此时程序不会响应键盘、鼠标、菜单的操作,从而产生反应迟钝甚至死机,而使用多线程技术,将耗时的操作置于一个新的线程,可以避免这种尴尬的情况。

## 2 使用递归与多线程技术的原因

### 2.1 使用递归的理由

由于磁盘分区或文件目录是一个树型结构,是一

种递归的数据结构,所以对磁盘分区或文件目录的遍历就相当于对整个树型结构进行遍历,显然用递归算法是合适的,因为它符合使用递归算法的两个条件,一是通过反复调用自身来对整个树型结构进行遍历;二是有一个递归结束条件,即文件结束标志。通过判断文件结束标志来找到递归出口<sup>[5,6]</sup>。

本系统使用递归算法是基于以下几个原因:

(1)采用递归算法把一个大型的复杂的程序细分为小型的简单的程序,增加了程序的可读性,简化了代码,提高了工作效率。

(2)递归算法能够有效对树型结构中的每个结点进行遍历。

(3)符合使用递归算法的两个条件<sup>[7]</sup>。

### 2.2 系统使用多线程技术的理由

(1)在查找丢失文件信息时需要遍历硬盘中某分区或目录中的文件信息,被查找分区或者目录中丢失文件信息和现存文件信息,文件数量很大,消耗多。

(2)在遍历分区或目录时被找到的丢失文件数量巨大,把丢失文件各种信息写入丢失文件列表需要大量时间。

表 1 主要数据结构

变量名	类型	访问方法	描述
fdtitem	struct rootsector { BYTE filename[9]; // 文件名 BYTE extendname[4]; // 扩展名 CTime deltime; // 被丢失时间 CTime createtime; // 创建时间 WORD highaddress; // 地址高 16 位 WORD lowaddress; // 地址低 16 位 DWORD length; // 大小 struct rootsector * next; }	Public	各文件信息
* delfilepointer1	与上同	Public	丢失文件各信息(链表头)
* delfilepointer2	与上同	Public	丢失文件各信息(某结点)
* existfilepointer1	struct existfileinfo { CTime createtime; // 创建时间 WORD highaddress; // 地址高 16 位 WORD lowaddress; // 地址低 16 位 DWORD length; // 大小 struct existfileinfo * next; }	Public	现存文件各信息(链表头)
* existfilepointer2	与上同	Public	现存文件个信息(某结点)

(3)在遍历分区或目录过程中被找到的丢失文件和现存文件信息数量同样很巨大,系统需要动态分配内存来保存找到的数量巨大的丢失文件信息和现存文件的信息,这样同样需要大量时间<sup>[8,9]</sup>。

(4)用户机器硬件差异,使用多线程技术能有效地提高工作效率,避免时间浪费。

(5)使用多线程技术可使系统对鼠标等操作作即时反应。避免产生反应迟钝和死机等现象。

综上所述:使用多线程技术能大大提高系统的工作效率,使用多线程技术是必要的。

### 3 查找设计

本设计以 FAT32 文件系统为例,以 Visual C++ 6.0 作为开发工具,以 Windows2000/NT/XP 为应用平台进行设计。

#### 3.1 主要数据结构设计

根据操作系统对文件的管理组织形式,采用了单链表的结构,其中有两个重要链表,一是丢失文件的信息链表;二是现存文件的信息链表。主要包含信息如表 1 所示。

#### 3.2 算法设计

查找涉及到对整个硬盘分区或分区目录进行遍历,遍历过程中系统会对找到的丢失文件和现存文件分配内存空间,所以当遍历过程中文件的数量非常庞大时,系统将消耗很大的内存空间和占用较高的 CPU 资源进行处理<sup>[10]</sup>。因此,必须尽可能地简化各种内部条件,精简各种不必要的开销,力求以最高的效率来实现对磁盘的遍历。系统在设计过程中独立开辟线程。它可以有效避免由于文件数量巨大而发生的死机或者系统反应迟钝的现象。

最核心的设计是对 FAT32 硬盘分区或其分区中目录进行递归搜索,在递归过程中创建丢失文件链表及现存文件链表。当需要对某个丢失文件进行数据恢复时,可以从丢失文件链表中找到丢失文件信息,在现存文件信息链表中查找与丢失文件创建时间最接近的现存文件,参照其地址高 16 位,确定丢失文件地址范围。可通过定位到该文件地址并判断文件头标志和读取文件大小长度的数据来对该文件进行恢复<sup>[11,12]</sup>。这里主要讨论查找部分,查找丢失文件的流程图如图 1 所示。

具体设计方法如下:

(1)打开丢失文件所在磁盘分区。

```
HANDLE CreateFile ( LPCTSTR lpFileName,
DWORD dwDesiredAccess, DWORD dwShareMode,
LPSECURITY_ATTRIBUTES lpSecurityAttributes,
```

```
DWORD dwCreationDisposition, DWORD dwFlagsAndAttributes, HANDLE hTemplateFile);
```

(2)查找引导扇区中 BPB 即磁盘参数块中关于该磁盘的一些基本信息。

```
DWORD SetFilePointer ( HANDLE hFile, LONG
lDistanceToMove,
```

```
PLONG lpDistanceToMoveHigh, DWORD dw-
MoveMethod);
```

```
BOOL ReadFile(HANDLE hFile,BYTE * lpBuffer,
LONG nNumberOfBytesToRead, LONG lpNum-
berOfBytesRead,NULL);
```

(3)BPB 中计算出数据区根目录开始处。

(4)从根目录进入到需要查找的目的地的地址。

(5)从该地址开始读取目的地所有丢失文件、所有现存文件,包括文件夹中文件的目录项信息,判别出有效丢失文件的信息。

(6)把丢失文件信息显示在系统列表框中。丢失文件信息和现存文件信息写入丢失文件链表和现存文件链表中。

具体实现如图 1(查找丢失文件流程)所示:对目录项第一字节进行判断,当第一字节为 E5 时,该目录项是丢失文件目录项,则把丢失文件信息写入丢失文件链表并显示在列表框中;当第一字节为 00 时,该目录项是空,则表示目录项已经结束;当第一字节为 2E 时,表示该目录项为本目录的目录项,则跳出继续扫描。当第一字节为其他时,表示该目录项为现存文件目录项,则写入现存文件链表。

### 4 实验结果及分析

通过黑盒法进行功能测试,系统实现了对用 shift + del 文件删除、回收站文件删除的快速查找。

用户根据需要可选择查找类型:按盘区查找、目录查找、文件名查找、扩展名查找、文件全名查找、文件删除日期查找、文件大小查找等,系统根据用户选择类型和输入信息返回查找结果消息。但当查找范围大且所在范围的文件数量非常庞大时,或文件碎片太多时,系统会占用较多的内存空间和 CPU 时间,查找效率有待加强。

### 5 结束语

根据操作系统对文件的管理组织形式及丢失文件的查找情形,在分析递归算法及多线程的基础上,使用递归方法与多线程技术查找丢失文件,设计了丢失文件查找算法,实现对显示在列表框中所有丢失文件信息的定位,帮助用户快速找到需要恢复的丢失文件。

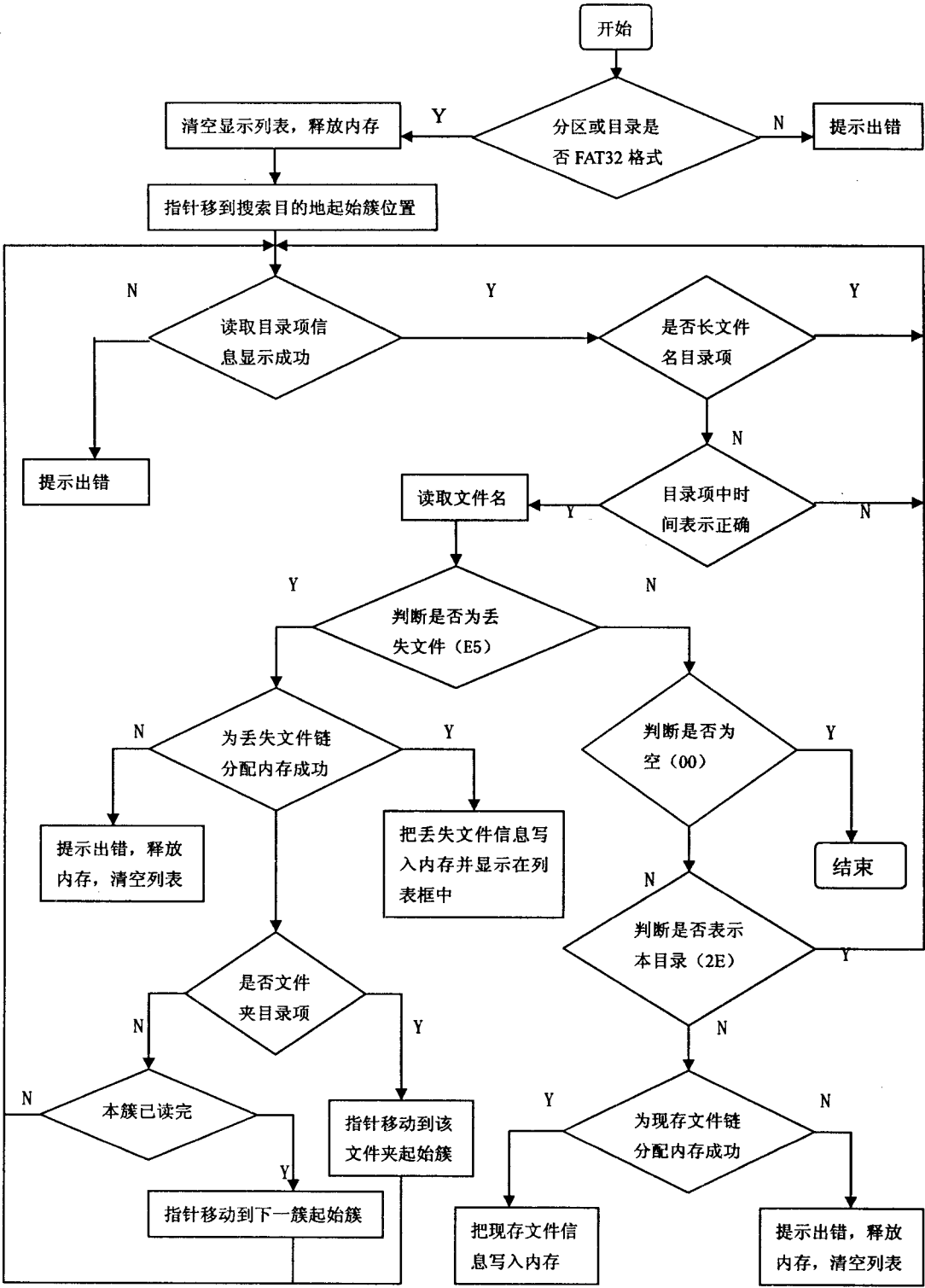


图 1 查找丢失文件流程图

参考文献:

[1] 辛 玲,王相海.国际象棋中马的周游路线问题的递归算法[J].计算机工程与设计,2006,27(1):47-48.

[2] 王 华,马 亮,顾 明.线程池技术研究与应用[J].计算机应用研究,2005(11):141-142.

[3] Van Baar R B, Alink W, Van Ballegooij A R. Forensics memory analysis: File mapped in memory[J]. Digital Investigation, 2008(5):52-57.

[4] Luoma V M. Computer forensics and electronic discovery: The new management challenge[J]. Computer & Security, 2006 (25):91-96.

[5] 游春晖,刘乃琦,代立松.数据恢复技术在计算机取证系统中的应用[J].成都大学学报,2008(6):131-133.

反走样算法只能适用于单像素宽的直线绘制的不足,文中算法可以适用于任何宽度的直线,并且都能取得很好的显示效果,同时不仅对单一背景色,对复杂背景色也都能获得平滑的绘制效果,这是其他算法不能做到的。

在实验中也对算法的效率做了相关的测试,在显示分辨率为  $240 \times 320$  屏幕上,随机产生 100 条直线并进行反走样绘制,基本整体无延迟刷屏效果出现,非常适合于嵌入式设备上的直线反走样绘制。



图5 多背景色下整体效果

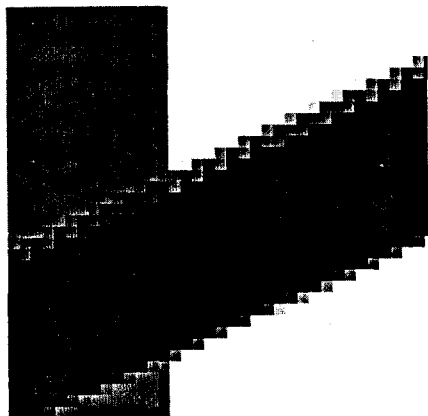


图6 多背景色下局部放大效果

## 5 结束语

文中在 DDA 直线光栅算法和 Wu 反走样算法的基础上进行了扩展,给出了一种快速的可以支持任意宽度的直线反走样绘制算法,并且能适应复杂背景颜色下的绘制,并优化去掉算法中的浮点运算,提高了算法的执行效率,经过实验证明,非常适合于嵌入式设备

上的美观性反走样直线的绘制。

## 参考文献:

- [1] 杭俊,付勇.一种基于加权区域采样的直线反走样生成算法[J].计算机技术与发展,2009,19(6):138-142.
- [2] 杜晨辉,经亚枝.全罗盘画面反走样算法的研究和实现[J].南京航空航天大学学报,2002,34(4):391-393.
- [3] 张波,张焕春,经亚枝.罗盘刻度线反走样快速绘制算法的改进研究[J].计算机辅助设计与图形学学报,2003,15(1):71-75.
- [4] 江修,张焕春,经亚枝.三像素宽反走样直线的绘制算法研究[J].南京航空航天大学学报,2003,35(2):148-151.
- [5] 韩丽,唐棣.一个快速有效的直线反走样算法[J].小型微型计算机系统,2005,26(3):509-511.
- [6] 李震霄,何援军.任意宽度直线的绘制与反走样[J].武汉大学学报:工学版,2006,39(4):130-133.
- [7] 谢莹,许荣斌,赵宏坤.基于嵌入式图形系统的改进 Bresenham 反走样算法[J].计算机技术与发展,2006,16(11):100-103.
- [8] 王宇平.光栅显示图形中直线反走样技术算法的改进[J].哈尔滨理工大学学报,2008,13(3):51-54.
- [9] 杨爱良,杨睿,熊智勇.反走样技术在计算机图形仿真中的运用[J].计算机仿真,2005,22(4):124-126.
- [10] 郝鑫,何援军,柳伟.一种基于灰度的三角形反走样算法[J].自动化技术与计算机技术,2008,25(5):143-146.
- [11] 沈强,张波,陈淑珍,等.计算机图形学反走样技术及实现[J].武汉大学学报,1997,43(1):113-118.
- [12] 孙家光,杨长贵.计算机图形学[M].北京:清华大学出版社,1995.
- [13] Wu X. An efficient anti-aliasing technique[J]. Computer Graphics, 1991, 25(4):143-152.
- [14] Diakopoulos N A, Stephenson P D. Anti-Aliased Lines Using Run-Masks[J]. Computer Graphics, 2005, 24(2):165-172.
- [15] Liu Yong Kui. An All-integer Algorithm for Drawing Anti-aliased Straight[J]. Computer Graphics, 1994, 13(4):219-222.
- [16] Thomas D, Netravali A N, Fox D S. Anti-aliased Ray Tracing with Covers[J]. Computer Graphics, 1989, 8(4):325-336.

(上接第 101 页)

- [6] Morgan T D. Recovering deleted data from the windows registry[J]. Digital Investigation, 2008(5):33-41.
- [7] 崔蓓蓓,陶亮.并行格型结构实现基于 DCT 的 2D 实值离散 Gabor 变换[J].计算机技术与发展,2009,19(2):105-108.
- [8] 王硕,尤枫,赵恒永.网页树型结构快速加载大数据量数据的实现[J].计算机工程与应用,2008,44(27):166-171.

- [9] 杨祖龙,吴国平.递归算法在树型视图中的应用[J].计算机工程,2002,11(3):139-141.
- [10] 赵亚萍.基于 Visual C#.NET 的空间缓冲区分析开发[J].计算机技术与发展,2009,19(12):29-35.
- [11] 杨泽明,许榕生,刘宝旭.文件删除的恢复与反恢复[J].信息安全,2002(4):38-41.
- [12] 涂彦晖,戴士剑.数据安全与编程技术[M].北京:清华大学出版社,2005:30-50.