1. **需求分析**
   1. 程序任务

本程序的任务是以一个二进制文件为文件卷，建立一个基于Windows系统的二级文件系统。用户使用可执行程序，可以输入程序提供的指令进行文件管理操作，具体包括创建文件、打开文件、读取文件、写入文件、删除文件、定位文件指针、列目录、格式化文件卷等。相应的输出形式为文件句柄、读取写入字节数等结果。

1. **概要设计**

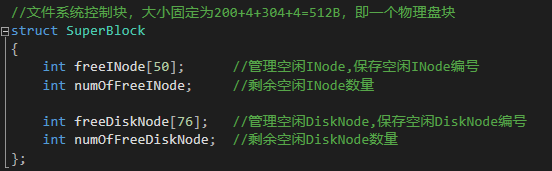
**2.1 任务分解**

设计二级文件系统的任务主要可以分解为磁盘结构定义、Superblock、inode和Disknode的数据结构定义、文件操作接口定义、文件操作接口实现以及用户交互设计。

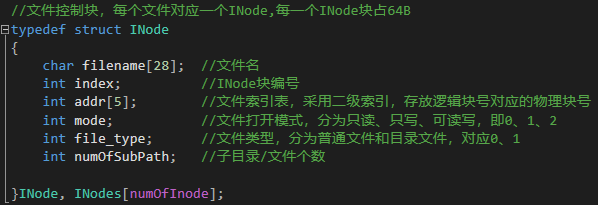
**2.2 数据类型**

本程序采用C++语言实现，将磁盘Superblock、inode和Disknode全部采用结构体实现。整个虚拟文件卷共占87个物理盘块，Superblock占1个，INode区占10个，DiskNode区占76个。

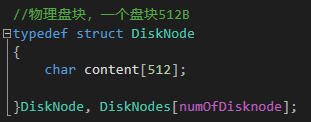
Superblock的结构体定义如下图：



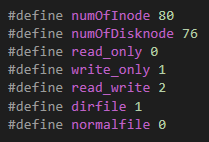
Inode的结构体定义如下图：



Disknode的结构体定义如下图：



除了上述三个结构体，本程序还定义了如文件类型、文件访问权限等宏定义如下图所示：



**2.3 主程序流程**

主程序流程图如下：

当前目录是否存在disk文件

进入用户交互界面，等待用户输入，返回结果

Y

读取disk,初始化相关数据结构

N

创建disk文件

**2.4 模块间调用关系**

单向箭头表示出射模块调用了入射模块。

主模块是main函数，在main函数中首先调用InitFileSystem模块从disk文件卷中读取数据，赋值给对应数据结构。随后调用InputCommand模块与用户进行交互，执行用户要求的命令。用户要求的命令包括创建文件，打开文件，读取文件，写文件，移动文件指针和删除文件等。创建文件和打开文件都会调用SearchPath这个核心模块。删除文件会调用RetrieveINode模块和RetrieveDiskNode模块来回收Inode节点和文件占用的物理盘块

closeFile

RetrieveINode

RetrieveMemory

OpenFile

createFile

deleteFile

ls

SearchPath

writeFile

readFile

InputCommand

InitFileSystem

Main

AllocMemory

AllocINode

1. **详细设计**

**3.1 重点函数**

(1) int SearchPath(string filename)

参数为文件名，若文件存在，则返回文件的INode节点的编号(从1开始)。

函数首先使用函数对文件名以/为分隔符进行分割，得到各级目录文件的文件名。以根目录root为起点(根目录文件的INode编号固定为1)，以类似UNIXV6++的目录检索方式，以32个字节为一个目录项，前28字节为文件名，后4个字节为对应文件的INode编号。这样一级一级地检索，直到找到文件的INode节点。如果路径存在错误或者找不到该文件，则函数返回-1。

对路径中的每一个文件名

在其父目录文件中进行检索

Y

是否检索到

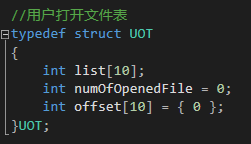
N

返回-1

(2) int createFile(string filename, int file\_type, int mode)

该函数用于创建文件。参数为文件名，文件类型和文件读写权限，返回值为文件INode节点的编号，同时也是文件句柄。函数首先**调用SearchPath**模块搜索文件是否已经存在，如果文件已经存在则返回文件的INode节点编号。如果文件不存在，再检查是否还有空闲INode节点，如果没有则返回-1。否则**再次调用SearchFile**模块找到父目录的INode节点，根据父目录INode节点的变量确定新目录项插入的位置，然后将新申请的INode节点编号和文件名存入目录项。如果父目录已经达到最大文件数量，则无法创建，返回-1。

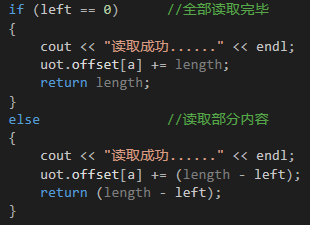
(3) int openFile(string filename, int mode)

该函数用于打开文件。参数为文件名以及文件打开方式。函数首先调用SearchPath模块查找文件对应的INode编号，如果文件不存在或者打开方式不符合权限则返回-1。否则在用户文件打开表中查找文件是否已经被打开，如果已打开直接返回句柄，否则更新UOT，再返回。

(4) int readFile(int index, char\* buffer, int length)

该函数用于从文件读取数据到指定数组。参数为文件句柄(INode节点编号)，数组首地址以及欲读取的字节数。返回值为实际读取的字节数。函数首先检查用户打开文件表，查看文件是否已经打开。如果未打开，则返回-1.如果已打开，则检查文件是否为只读或者可读写文件，如果不是则返回-1。如果已打开且满足读写权限，则根据该文件文件指针确定读起始地址，定义相关变量记录剩余待读取字节数、数组偏移等。读取完毕后，更新文件指针位置，返回实际读取的字节数。

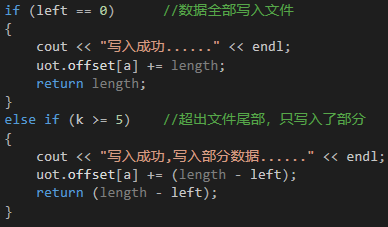




(5) int writeFile(int index, const char\* buffer, int length)

该函数用于从指定数组写入length字节到文件中。参数为文件句柄，数组首地址以及欲写入的字节数。函数首先检查用户打开文件表，查看文件是否已经打开。如果未打开，则返回-1.如果已打开，则检查文件是否为只写或者可读写文件，如果不是则返回-1。如果已打开且满足读写权限，则根据该文件文件指针确定写起始地址，定义相关变量记录剩余待写入字节数、数组偏移等。写入完毕后，更新文件指针位置，返回实际写入的字节数。实际写入的字节数可能因为文件剩余磁盘空间不足而小于length。





(6) int seekPos(int index, int offset)

该函数用于定位文件读写指针。参数为文件句柄以及文件读写指针偏移量。函数首先查看用户打开文件表，如果文件已打开，则直接修改读写指针位置，返回1表示成功。如果未打开，则返回0表示失败。

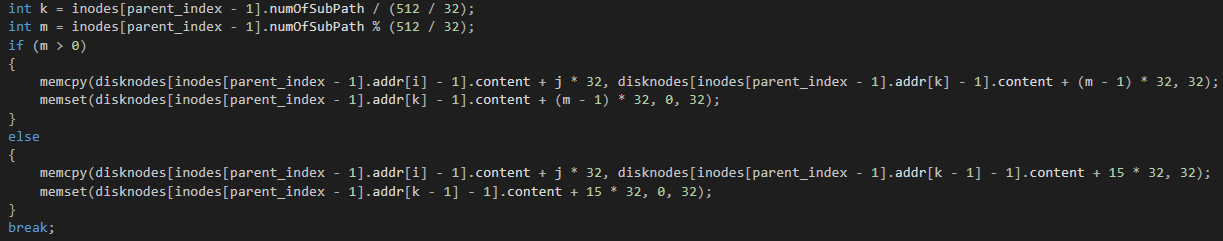
(7) bool deleteFile(string filename)

该函数用于删除指定文件。参数为文件名(包含完整路径)。函数首先**调用SearchPath模块**查找文件名对应的INode节点，如果文件不存在，则返回false。如果文件存在，则**调用RetrieveDiskNode模块**回收物理盘块，**调用RetrieveINode模块**回收INode节点。接着**再次调用SearchPath模块**查找文件父目录文件INode节点，接着从父目录文件的第一个物理盘块开始以32个字节为单位取出目录项进行文件名的匹配，如果匹配成功则将最后一个目录项的32字节内容覆盖到该32字节上，然后将原本最后一个目录项的32字节设置为全0。

查找父目录INode:



更新父目录：



**3.2 函数间调用关系**

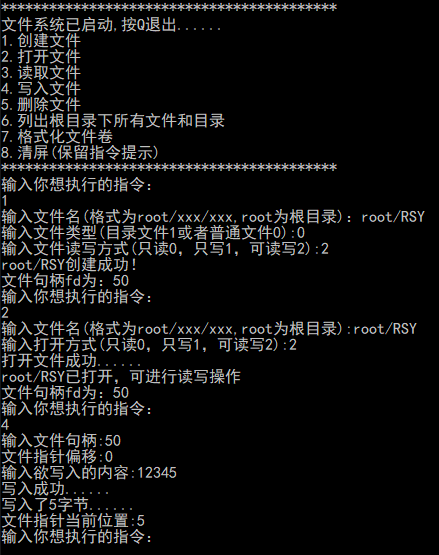
函数间调用关系类似于模块间调用关系，见概要设计。

1. **调试分析**

**4.1 测试及测试结果**

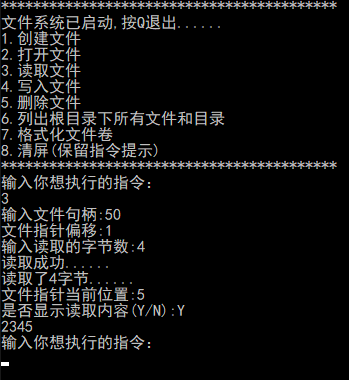
(1) 测试数据：在根目录root下创建RSY普通文件，从文件开头写入字符串12345，共5个字节。

测试结果：



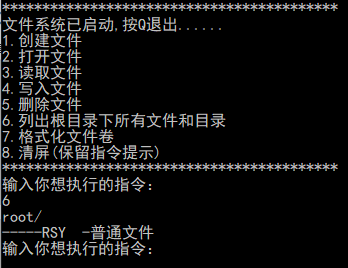
(2) 测试数据：从RSY文件的第二个字节开始，读取4个字节的内容。

测试结果：



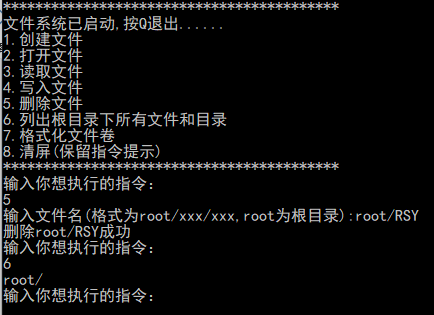
(3) 测试数据：列目录。

测试结果：



(4) 测试数据：删除RSY文件，然后列目录。

测试结果：



**4.2 复杂度分析**

程序中涉及的复杂度分析主要是针对SearchPath模块，INode节点和DiskNode节点的分配与回收。

SearchPath模块中，假设文件路径长度为n，那么根据其搜索时的线性性质，时间复杂度为O(n)。

INode节点和DiskNode的分配与回收算法采用的方式是栈。通过出栈和入栈来进行节点的分配与回收，所以时间复杂度为O(1)。

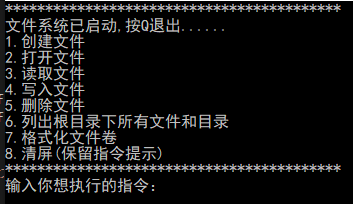
**4.3 问题思考**

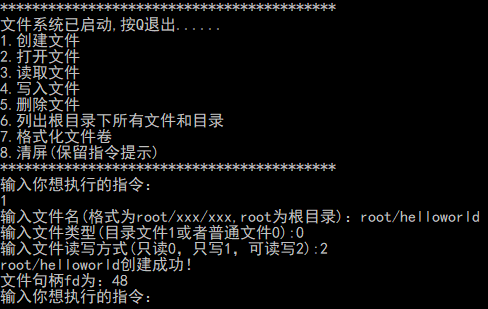
1. 在设计SearchPath模块时，需要将完整的文件路径进行拆分，拆分成从root根目录文件到文件本身的各级目录文件但，C++本身没有库函数能够根据指定分隔符拆分字符串。问题的解决是通过在网上查阅资料，查到利用C++标准库的vector容器和string类实现分割的函数，并加以利用。

2. 在设计deleteFile模块时，需要更新删除文件父目录文件的目录项。原本的思路是将删除文件的目录项后的其他剩余目录项依次向前覆盖，但实现时发现这样的思路太过复杂而且费时，代码冗长。问题的解决是我使用了另一种思路，即直接通过在目录文件INode节点中添加numOfSubpath变量记录当前包含了多少个子文件，然后直接将最后一个目录项的内容覆盖到删除文件的目录项中，再将原本最后一个目录项全置为0。

1. **用户使用说明**

本程序开发平台为Win10操作系统，可运行在Windows系统之上。本程序以命令行界面为交互界面，用户启动可执行文件之后会进入命令行界面。根据界面上的操作提示，用户输入想要执行的操作编号，根据提示进行输入。界面如下：





1. **实验总结**

这次操作系统课程设计实验的目标是设计实现一个二级文件系统。在经过了一个学期的操作系统课程的学习，我了解了UNIXV6++操作系统的文件系统是如何设计的。所以整个二级文件系统的思路与UNIXV6++的文件系统类似。实验一开始也遇到一些问题，比如不知道该如何利用Windows系统上的一个二进制文件作为虚拟文件卷，后来通过查阅、回忆之前C++课程中的文件相关操作和内存管理函数，解决了这个问题。在具体实现创建、打开以及读写等文件操作接口时反而并没有遇到太大的困难，更多的是需要考虑到所有的特殊情况。系统设计完成之后，自我感觉还不错，因为整个系统1000多行的代码基本上全是自己写的，仅有一个字符串分割函数是参考网络上的资料，其他模块和功能全部是自己实现的，并且经过测试都能正确运行。总的来说，这次课程设计让我对文件系统有了更透彻的认识，也提高了代码能力。

1. **参考文献**

[1]. <https://www.cnblogs.com/carsonzhu/p/5859552.html> string字符串分割