**文件系统**

1152703 方志晗 软工2班

**解决问题：**模拟windows xp风格建立一个虚拟的文件系统。模拟完成文件的基本操作。

**设计思想与算法：**利用多路树模拟文件系统的目录（即文件夹）和文件的关系，通过多路树完成文件夹的递归嵌套。通过当前目录的切换可查看文件夹的内部信息，通过递归算法可以访问整个文件系统，从而计算整个文件系统的文件数目以及文件大小，从而可以算出当前系统的内存使用情况。利用递归调用复制功能可以完成整个文件夹机器内部链式信息的复制。通过粘贴复本的复本，可以使剪贴板上的复本一直存在直到下次被替换。若只是粘贴复本，只能粘贴一次。通过递归算法可以动态监测文件内部信息，并返回显示。通过丢弃并新建真个指针项可以达到格式化的功能。通过将不需要的信息放到空闲位置并重排实现删除重组的效果。通过设置标志位select可以找到当前目录下被选中的文件。通过设置前驱和后继指针完成前进后退功能。通过在每个目录下设置索引结合递归查找完成通过索引直接访问。

当然，最基本的是将坐标切分成条状正好适合文件的新建与删除。

主线程通过对标志位（打开，新建，删除，复制，粘贴等）的监听完成相应功能。

**数据结构：**

文件夹（文件目录）：

|  |
| --- |
| **public** FilePanel myFilePanel=**new** FilePanel();      //是否删除  **public** **boolean** isDelete =**false**;  **public** **int** FileNum=0;  **public** **int** FileVolume =0;  **public** String SocketName = "新建文件夹";  **public** **boolean** isSelect = **false**;    //文件夹内部信息项  **public** FilePanel myFilePanel=**new** FilePanel();  **public** Icon ImgIcon= **new** ImageIcon("filesocket.png");  **public** Icon zoneIcon = **new** ImageIcon("zone.png");      **public** **boolean** isZone =**false**; //是否为分区，如果是，另行处理  **public** **boolean** isOpen =**false**; //是否打开    //图标  **public** JLabel FileSocketLabel = **new** JLabel(**this**.ImgIcon);  //文件名  **public** JLabel FileSocketName =**new** JLabel("新建文件夹");  //文件数目  **public** JLabel FileNumInSocket =**new** JLabel(""+FileNum);  //文件大小  **public** JLabel FileVolumeInSocket = **new** JLabel(""+**this**.FileVolume+"B"); |

文件夹内部信息项：

|  |
| --- |
| //已有文件数目  **public** **int** filecount = 0;  **public** **int** socketcount=0; //已有的文件夹数目  **public** **int** allcount =0; //总数目  **public** **int** volum = 0; //文件夹包含的总内存      **public** **int** Scount=0; //对建立过的文件夹计数  **public** **int** Fcount=0; //对建立的文件计数  **public** **int** Ccount=0; //对复制信息计数    **public** AFileSocket myFileSocket=**null**; //文件夹目录    **public** String Instruction="\\\\root"; //文件夹索引，默认为根目录，建立时自动改变  **public** **static** **final** **int** *FILEMAXNUM*=10; //允许建立的最大文件数  **public** **static** **final** **int** *SOCKETMAXNUM* =10; //允许建立的最大文件夹数  //AFileSocket temp =null;  **public** FilePanel lastSocket =**null**; //记录上上一级文件夹  **public** FilePanel foreSocket =**null**; //记录下一级最近打开的文件夹，方便前进与后退      **public** AFileSocket HoldSocket[] = **new** AFileSocket[*SOCKETMAXNUM*]; //包含的文件夹数组  **public** TextFile HoldFile[]= **new** TextFile[*FILEMAXNUM*]; //包含的文件数组      //设置一个文件tree  **public** DefaultMutableTreeNode myFileTree = **null**; |

文件：

|  |
| --- |
| **public** FilePanel lastSocket=**null**;  **public** **int** FileVolume = 0;  **public** **boolean** isSelect =**false**;  **public** **boolean** isOpen = **true**;      FilePanel temp =**null**;  **public** Icon ImgIcon= **new** ImageIcon("file.png");  **public** String FileName ;  //图标  **public** JLabel FileLabel = **new** JLabel(**this**.ImgIcon);  //文件名  **public** JLabel FileNameLabel =**new** JLabel("");  //文件数目  **public** JLabel FileNumLabel =**new** JLabel(""+1);  //文件大小  **public** JLabel FileVolumeLabel = **new** JLabel(""+**this**.FileVolume+"B");    //保存文本信息  **public** String content="";  **public** **boolean** isDelete= **false**;  **public** **int** rootNum =0;  //树形图  **public** DefaultMutableTreeNode myFileTree = **null**; |

**功能说明：**



**新建文件夹：**在当前目录下建立一个新文件夹；

**新建文件：**在当前目录下建立一个新文件；

**删除：**选择一个当前目录下的文件或者文件夹后按此选项可以删除并重排目录下的所有文件；

**重命名：**选择一个当前目录下的文件或者文件夹后点击此按钮可以为选择的文件或者文件夹修改名称，自动检测重名，若重名，则命名失败；

**复制：**选择一个当前目录下的文件或者文件夹后点击此按钮可以将该文件或者文件夹复制到粘贴板，支持多次复制和跨文件夹，跨区复制，复制后自动更新文件夹（包含内部文件）或者文件信息。

**粘贴：**将已经复制到粘贴板的内容复制到当前目录；支持跨区，跨文件夹复制；

**格式化：**对分区可操作，点击格式化后，该分区的内存全部释放，内部所有文件和文件夹均消失；

**鼠标：**支持单机选择，选择后在左下文件信息目录显示当前选中文件或者文件夹的内部及内存占用信息，双击进入或者打开；

**自动更新当前系统和某个分区的文件使用状况；**

**文件树功能支持动态更新，点击选中文件所属的目录；**

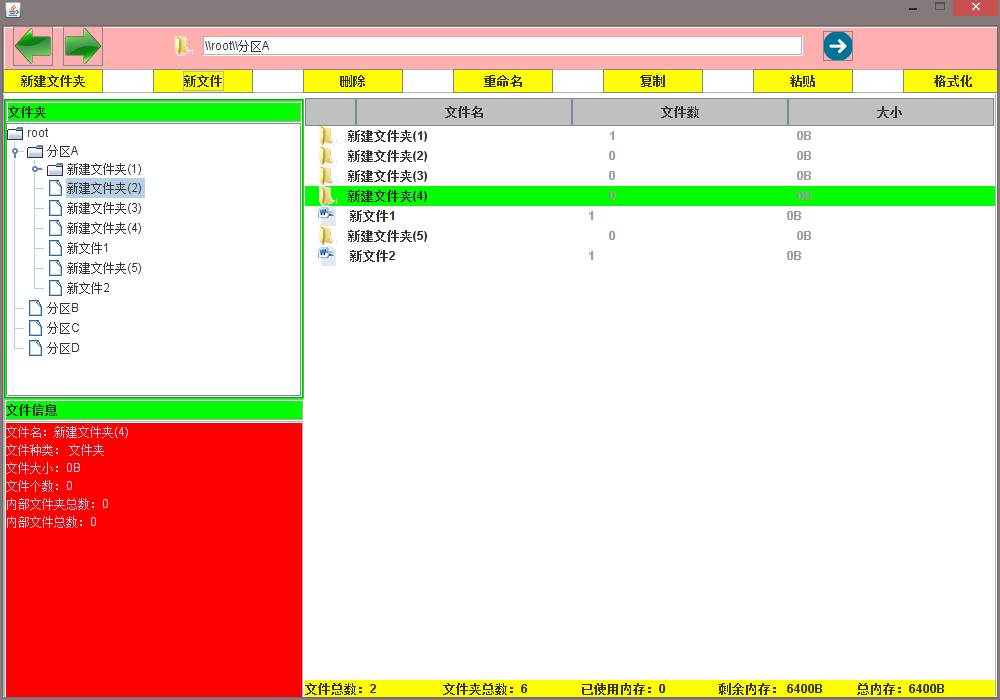
**后退/前进：**自动记忆访问顺序，可以支持前进和后退到之前访问的文件目录；

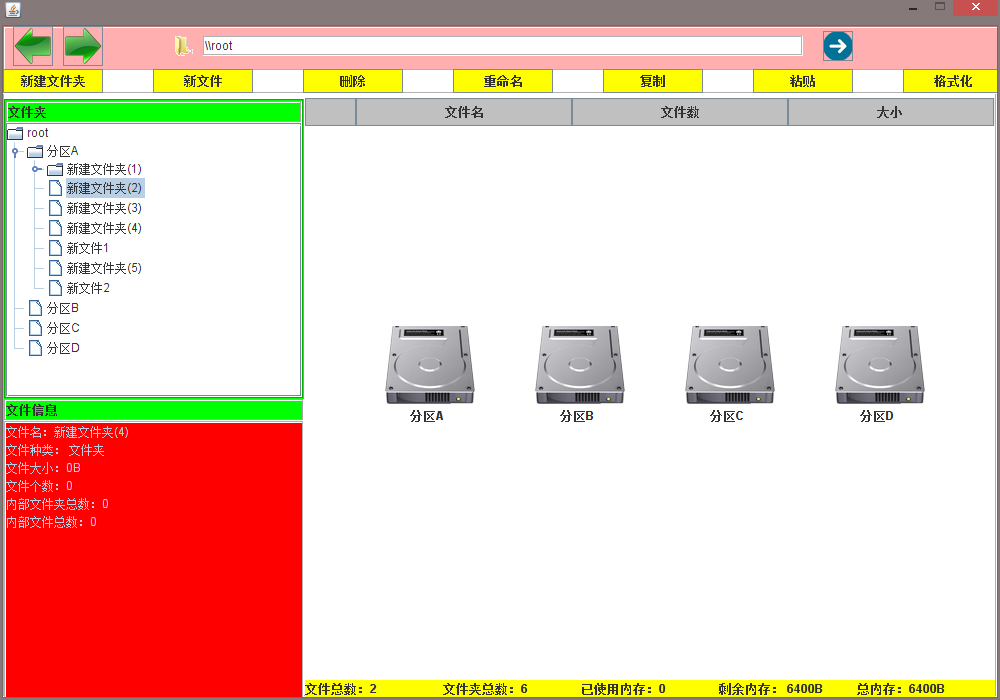
**转到：**直接在搜索框中输入文件或者文件夹的路径，点击转到可之间转到该目录；

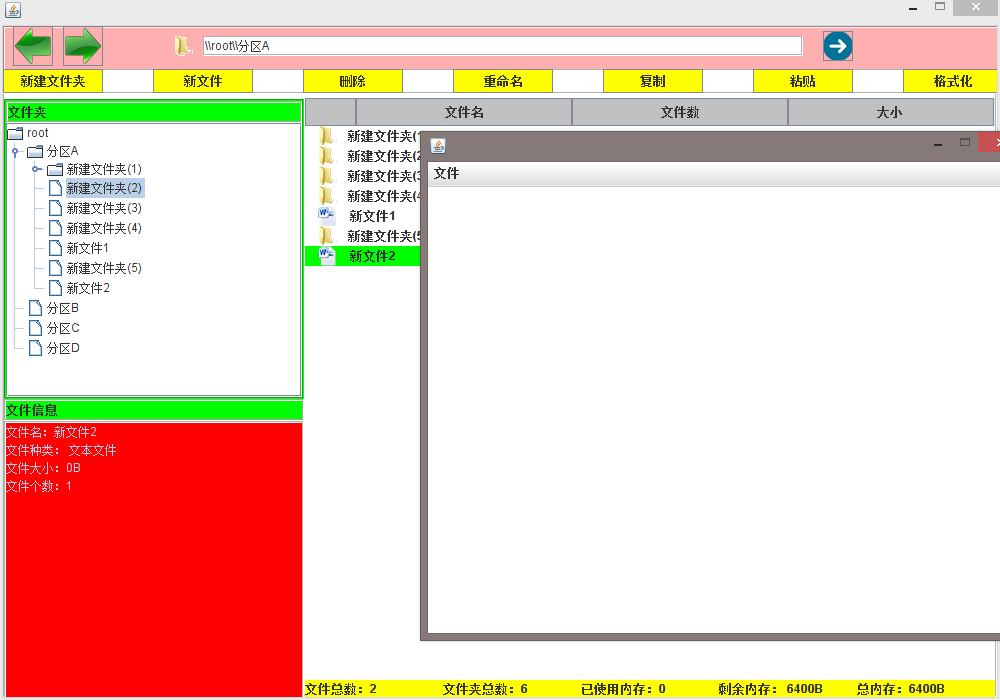
**搜索框：**支持搜索输入和动态更新到当前目录；

动态更新当前系统的信息，显示在右下方；

**界面如下：**







关键算法：递归；

关键API：

|  |
| --- |
| //查找文件  **public** **void** SortFileSocket(FilePanel temp)  {    **this**.SortString=**this**.Introduction.getText();  **if**(temp!=**null**)  {  **if**(temp.Instruction.equals(SortString))  **this**.newPanel=temp;  **else**{  **for**(**int** i=0;i<temp.socketcount;i++)  { **if**(temp.HoldSocket[i]!=**null**)  {  **if**(temp.HoldSocket[i].myFilePanel.Instruction.equals(**this**.SortString))  {    **this**.newPanel =temp.HoldSocket[i].myFilePanel;  **break**;  }  **else**  {  SortFileSocket(temp.HoldSocket[i].myFilePanel);  }  }  }  }  }    } |

|  |
| --- |
| //重命名函数  **public** **void** Rename()  {    **for**(**int** i = 0;i<**this**.myFilePanel.filecount;i++)  {  **if**(**this**.myFilePanel.HoldFile[i].isSelect)  {  String inputValue = JOptionPane.*showInputDialog*("请输入文件名：");  **if**(**this**.myFilePanel.hasSameNameFile(inputValue,i))  {  JOptionPane.*showMessageDialog*(**null**, "命名重复：已存在\""+inputValue+"\"的文件", "重复命名", JOptionPane.*ERROR\_MESSAGE*);  }  **else**  {  **if**(inputValue!=**null**&&!inputValue.equals(""))  {  **this**.myFilePanel.HoldFile[i].Rename(inputValue);  }  }  }    }  **for**(**int** i = 0;i<**this**.myFilePanel.socketcount;i++)  {  **if**(**this**.myFilePanel.HoldSocket[i].isSelect)  {    String inputValue = JOptionPane.*showInputDialog*("请输入文件名：");  **if**(inputValue!=**null**&&!inputValue.equals(""))  {  **if**(**this**.myFilePanel.hasSameNameSocket(inputValue,i))  {  JOptionPane.*showMessageDialog*(**null**, "命名重复：已存在\""+inputValue+"\"的文件", "重复命名", JOptionPane.*ERROR\_MESSAGE*);  }  **else**  {  **this**.myFilePanel.HoldSocket[i].Rename(inputValue);  }  }  }        }  **this**.myFilePanel.repaint();  } |

|  |
| --- |
| //进入文件夹  **public** **void** changeFilePanel(FilePanel NewPanel)  {  **this**.CenterPanel.remove(**this**.myFilePanel);  **this**.CenterPanel.repaint();  //this.myFilePanel=null;  **this**.myFilePanel=NewPanel;  **this**.CenterPanel.add(**this**.myFilePanel,BorderLayout.*CENTER*);  **this**.myFilePanel.repaint();        //复制  **public** AFileSocket getACopy()  {  AFileSocket temp1 = **new** AFileSocket(**this**.SocketName,**this**.rootNum);  temp1.FileNum=**this**.FileNum;  temp1.FileVolume=**this**.FileVolume;  temp1.myFilePanel.Instruction=**this**.myFilePanel.Instruction;  **for**(**int** i=0;i<**this**.myFilePanel.socketcount;i++)  {  **if**(temp1.myFilePanel.socketcount<FilePanel.*SOCKETMAXNUM*)  {  AFileSocket temp = **this**.myFilePanel.HoldSocket[i].getACopy();    //该文件夹下创建文件夹的上一级为该文件夹  temp.myFilePanel.lastSocket=temp1.myFilePanel;  temp1.myFilePanel.HoldSocket[temp1.myFilePanel.socketcount++] = temp;  temp.myFilePanel.setInstruction();      temp1.myFilePanel.addNewFileSocket(temp1.myFilePanel.HoldSocket[temp1.myFilePanel.socketcount-1], temp1.myFilePanel.allcount);  temp1.myFilePanel.Scount++;  temp1.myFilePanel.allcount++;  //MainPanel.WholeFileSocketNum.setText("文件夹总数："+(++MainPanel.WholeSocketNumCount));  **if**(temp1.myFilePanel.myFileSocket!=**null**)  temp1.myFilePanel.myFileSocket.updateFileSocket();  }    }    **for**(**int** i=0;i<**this**.myFilePanel.filecount;i++)  {  **if**(temp1.myFilePanel.filecount<FilePanel.*FILEMAXNUM*)  {  TextFile temp = **this**.myFilePanel.HoldFile[i].getACopy();    //该文件夹下创建文件夹的上一级为该文件夹  temp.lastSocket=temp1.myFilePanel;  temp1.myFilePanel.HoldFile[temp1.myFilePanel.filecount++] = temp;  //this.HoldSocket[this.socketcount++] = new AFileSocket("新建文件夹"+(1+this.filecount++));    temp1.myFilePanel.addNewFile(temp1.myFilePanel.HoldFile[temp1.myFilePanel.filecount-1], temp1.myFilePanel.allcount);  //System.out.println(212);  temp1.myFilePanel.Fcount++;  temp1.myFilePanel.allcount++;  //总文件数增加  //MainPanel.WholeFileNum.setText("文件总数："+(++MainPanel.WholeFileNumCount));  **if**(temp1.myFilePanel.myFileSocket!=**null**)  temp1.myFilePanel.myFileSocket.updateFileSocket();  }  }      **return** temp1;  } |

|  |
| --- |
| //粘贴时更新数据-->递归  **public** **void** flushDataWhenPaste(FilePanel root)  {      **if**(root.myFileSocket!=**null**)  {  root.myFileSocket.FileNumInSocket.setText(""+root.myFileSocket.FileNum);  root.myFileSocket.FileVolumeInSocket.setText(""+root.myFileSocket.FileVolume+"B");        // root.myFilePanel.setInstruction();  **if**(root.myFileSocket.myFilePanel.lastSocket!=**null**)  root.Instruction=root.lastSocket.Instruction+"\\\\"+root.myFileSocket.SocketName;  **for**(**int** i=0;i<root.socketcount;i++)  {    flushDataWhenPaste(root.HoldSocket[i].myFilePanel);  }  }        } |

|  |
| --- |
| //鼠标监听  **if**(e.getButton()==e.*BUTTON1*)  {  //设置选择的暗影  **this**.isSelect=!**this**.isSelect;  **if**(**this**.isSelect)  {  **this**.setBackground(Color.*green*);  **this**.flushAll();  }  //两次则进入  **if**(!**this**.isSelect)  {  **this**.setBackground(Color.*white*);  **this**.isOpen =**true**;    }  //其他文件取消选择  **for**(**int** i=0;i<**this**.myFilePanel.lastSocket.socketcount;i++)  {  **if**(**this**.myFilePanel.lastSocket.HoldSocket[i]!=**this**)  {  **this**.myFilePanel.lastSocket.HoldSocket[i].isSelect=**false**;  **this**.myFilePanel.lastSocket.HoldSocket[i].setBackground(Color.*white*);  }  }  **for**(**int** i=0;i<**this**.myFilePanel.lastSocket.filecount;i++)  {    **this**.myFilePanel.lastSocket.HoldFile[i].isSelect=**false**;  **this**.myFilePanel.lastSocket.HoldFile[i].setBackground(Color.*white*);    }      } |