菜单的深度优先遍历在 C#中的实现及应用

Implementation and Application of Visiting Menu Depth – First in C#

娄七明　赵东晋　傅锦伟　许海成　（红河学院工学院 云南蒙自 661100）

摘　要： 在 MIS 应用中，通常要对用户权限进行控制。本文首先分析了两种常用的访问控制方法存在的缺陷；然后提出了一种基于深度优先遍历算法实现系统菜单遍历及用户权限控制的方法；最后结合实例用 C#实现了该方法。该方法具有很好的通用性和实用性。

关键词： 深度优先　队列　树　MIS

1　引言

在 MIS 系统中，为了保证系统的安全性，必须对用户权限进行控制。目前，最常用的方法是在程序中根据用户的身份进行权限逻辑判定[1]。然而，这种方法需要预先设计好权限分配逻辑，并在编程时实现，若某用户的权限需要改变，必须修改应用程序并重新编译。另一种改进方法是，将权限信息存储在数据库中，在需要动态改变用户权限时，通过更改存储的权限信息来实现权限控制[2]。但是，若系统需求发生变化，增加或删除某些应用程序的功能时，需要手工更改数据库中的权限信息，还可能需要对程序中的权限判断进行修改。如果能遍历应用程序的菜单，把遍历结果存入数据表中，然后再根据菜单分配用户权限，就能解决上述方法存在的缺陷，实现真正意义上的用户权限动态控制。

C#是. net 平台中一个非常优秀的基于数据库应用的开发工具，被广泛用于 MIS 系统开发。本文采用深度优先遍历算法实现了基于 C#的 MIS 系统菜单的自动遍历，并把遍历结果存入数据表中，然后根据用户权限设置菜单项的 Enabled 属性，实现用户权限的动态控制。该方法具有三大特点：①以类的形式提供，通用性好；②采用深度优先算法遍历，菜单项的从属关系明确；③可遍历菜单的级数达到 3 级，满足了大多数系统的需求。

2　算法分析

遍历菜单就是按一定的顺序访问每一个菜单项，并使每个菜单项只被访问一次。菜单可以看成是一棵 N 叉树，图 1 是某图书管理 MIS 的菜单结构。这里为了较好地描述算法，把图中第 1 层、第 2 层、第 3 层的对应菜单项分别记为 1 级、2 级、3 级菜单项，则深度优先遍历菜单的算法可描述如下：

① 首先定义并初始化队列 q1，用于存放遍历过的菜单项；

② 通过循环从左到右依次访问每个 1 级菜单项，把菜单项插入 q1，若被访问的 1 级菜单项有子菜单项，则转到③；否则，继续循环，循环结束时转到⑤；

③ 通过循环从上到下依次访问第②步中正在遍历的 1 级菜单项所包含的每个 2 级菜单项，把菜单项插入 q1，若被访问的菜单项有子菜单项转到④；否则，继续循环，循环结束时转入②继续；

④ 通过循环从上到下依次访问第③步中正在遍历的 2 级菜单项所包含的每个 3 级菜单项，把菜单项插入 q1，直到循环结束，转到③继续；

⑤ 遍历结束。

按照上述算法遍历图 1 所示的菜单，则遍历结束后队列 q1 的状态如图 2 所示。

3　算法分析

由上面的分析知，菜单的遍历结果需使用队列进行存放，下面给出链式队列在 C#中的实现。

3.1　队列结点的定义

为了很好地描述菜单项之间的从属关系，队列的每个结点都包含 data、n、pn 和 next 四个数据域，分别用于存放菜单项的名称、编号、父菜单项编号和指向下

一结点的指针。结点的定义如下：

```
public class CNode
{
public ToolStripMenuItem data;//菜单项
public int n,pn;//n 菜单项编号,pn 父菜单项编号
public CNode next;//指向下一结点的指针
public CNode( ) {next = null;} //构造函数
}
```

其中,CNode( )是构造函数,把结点的 next 域初始化为 null。

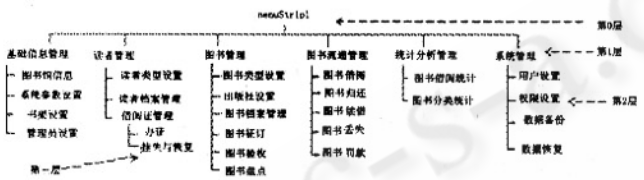

图 1　图书管理系统菜单结构

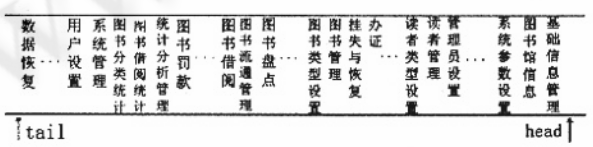

图 2　队列 q1 的状态

3.2 链式队列的定义及实现

队列有队首和队尾,因此链式队列应该设有队首指针(head)和队尾指针(tail),分别指向队首元素和队尾元素。队列有如下基本操作：

① 队列的初始化,就是建立一个空队列,即把队首指针和队尾指针赋值为 null;

② 入队,就是往队列中插入一个元素。如果队列为空,只需把队尾指针和队首指针都指向待插入的元素;如果队列不为空,则要把队尾元素的 next 域指向待插入的元素,并修改新插入节点的 next 域为 null;

③ 出队,就是返回队首元素。如果队列为空,不进行如何操作;如果队列中只有一个元素,返回队首指针的值,并设置队首指针和队尾指针的值为 null;如果队列中有多个元素,返回队首指针的值,并设置队首指针指向下一个元素;

④ 判断队列是否为空。链式队列为空的条件是队首指针的值为 null。

根据上面的分析,链式队列用 C#实现如下：

```
public class QueueCnode
{private CNode head,tail;
/*head 是队首指针,tail 是队尾指针*/
public QueueCnode( ){head = tail = null;}
/*构造函数,实现队列的初始化*/
public void Insert(ToolStripMenuItem data,int x,int
px)
/*入队*/
        {
        if (head = =null)//队列为空
        {head = new CNode( );head. data = da-
    ta;
        head. n = x;head. pn = px;tail = head;}
        else//队列不为空
        {tail. next = new CNode( );tail. next. data
    = data;
tail. next. n = x; tail. next. pn = px; tail = tail.
next;}}
public void Remove(ref CNode ch)//出队操作
{ch = head;//把队列的首结点赋值给 ch
if(head = =null){}//队列为空
else if (head = =tail) //队列中只有一个结点
{head = tail = null;}
else{head = head. next;}}//队列中有多个结点
public bool IsEmpty//判断队列是否为空
{get{if (head = = null) return true; else return
false;}}}
```

其中,QueueCnode( )是构造函数,用于初始化队列;Insert 方法实现入队操作;Remove 方法实现出队操作;属性 IsEmpty 判断队列是否为空。

4　菜单的深度优先遍历实现

4.1 创建表存放菜单项[3]

为了实现 MIS 系统中用户权限的动态控制,需要在 MIS 系统的数据库中建立 tabmenu、tabuser、tabqx 三个表,分别存放菜单信息、用户信息、用户权限信息。三个表对应的关系模型如下：

tabmenu(menunum,menutext,parentmenunum)

tabuser(userid,username,pwd)

tabqx(userid,menunum,qx)

其中,表 tabmenu 中的字段 menunum、menutext、parentmenunum 分别表示菜单编号、菜单标题和父菜单编号,主码是 menunum;表 tabuser 中的字段 userid、username、pwd 分别表示用户编号、用户名和密码,主码是 userid;表 tabqx 中的字段 userid、menunum、qx 分别表示用户编号,菜单编号和对应的权限,主码是 userid 和 menunum。

4.2 算法实现

为提高算法的通用性和实用性,这里把菜单的遍历、菜单的存储以及用户权限的设置都封装到了同一个类 menusearch 中,其定义如下:

```
public class menusearch
{private MenuStrip mainmenu;//待遍历的系统菜单
private QueueCnode q1;//存放遍历结果的队列
private string constr;//数据库连接字符串
public menusearch(MenuStrip menu, string str);
public void Search();//深度优先遍历菜单
public void WriteData();//把遍历结果写入数据库
public void Setqx(int user);/*检索用户权限,设置相应菜单项的 Enabled 属性*/
}
```

其中,各成员函数的功能及实现如下:

① menusearch() 是构造函数,用于初始化成员 mainmenu、q1 和 constr。代码如下:

```
mainmenu = menu; constr = str; q1 = new QueueCnode();
```

② Search() 实现待遍历菜单的深度优先自动遍历,是整个算法的关键所在。根据前面的算法分析可知,要遍历如图 1 所示具有 4 个层次的菜单需使用三重循环才能实现。代码如下:

```
int x=1;//x 是一级菜单项的编号
foreach (ToolStripMenuItem m in mainmenu.Items)
/*遍历一级菜单项*/
{int y=0;q1.Insert(m,x,0); y=x*10+1;
/*y 是二级菜单项的编号*/
foreach (ToolStripMenuItem n in m.DropDownItems)
/*遍历二级菜单项*/
{q1.Insert(n,y,x);int z=0; z=y*10+1;
/*z 是三级菜单项的编号*/
foreach (ToolStripMenuItem h in n.DropDownItems)
/*遍历三级菜单项*/
{q1.Insert(h,z,y);z++;}y++;}x++;}
```

需要说明的是待遍历菜单的层次每增加一层,遍历菜单的代码只需在原来的基础上增加一层循环,即可实现待遍历菜单中各级菜单项的自动遍历。

③ WriteData() 把菜单的遍历结果写入表 tabmenu,以便在系统中按菜单项分配用户的操作权限。代码如下:

```
SqlConnection con=new SqlConnection(constr);
/*创建数据库连接对象*/
con.Open();//连接数据库
SqlCommand cmd=con.CreateCommand();
/*创建 SqlCommand 对象*/
/*以下代码设置 cmd 对象要执行的命令,并定义参数*/
cmd.CommandText="insert into tabmenu values(@mnum,@mtext,@pmnum)";
cmd.Parameters.Add("@mnum",SqlDbType.Int,4,"menunum");
cmd.Parameters.Add("@mtext",SqlDbType.VarChar,40,"menutext");
cmd.Parameters.Add("@pmnum",SqlDbType.Int,4,"parentmenunum");
CNode node1 = new CNode();
/*创建 Cnode 类型节点 node1*/
while(q1.IsEmpty==false)
/*如果存放菜单遍历结果的队列 q1 不为空*/
{q1.Remove(ref node1);
/*对 q1 进行出队操作,把出对的元素存入 node1*/
/*以下代码为 cmd 对象指定实参*/
cmd.Parameters[0].Value=node1.n;
cmd.Parameters[1].Value=node1.data.Text;
cmd.Parameters[2].Value=node1.pn;
cmd.ExecuteNonQuery();}
```

```
/ * 执行 SQL 命令把菜单项存入表 Tabmenu * /
con. Close( ) ;//关闭数据库连接
```

④ Setqx( ) 检索数据库中指定用户的操作权限，并设置相应菜单项的 Enabled 属性，从而实现用户权限的动态控制。代码如下：

```
SqlDataReader dr;
/ * 定义数据读取器 dr,用于读取数据 * /
SqlConnection con = new SqlConnection ( constr ) ;
con. Open( ) ;
SqlCommand cmd = con. CreateCommand( ) ;
cmd. CommandText = " select qx from tabqx where
userid = @ user and menunum = @ num" ;
cmd. Parameters. Add( " @ user" , SqlDbType. Int,
4," userid" ) ;
cmd. Parameters. Add( " @ num" , SqlDbType. Int,
4," menunum" ) ;
CNode node1 = new CNode( ) ;
while( q1. IsEmpty = = false)
{q1. Remove( ref node1) ;
cmd. Parameters[0]. Value = user; cmd. Parame-
ters[1]. Value = node1. n;
dr = cmd. ExecuteReader ( CommandBehavior. Sin-
gleRow) ;
/ * 执行 SQL 命令,把结果赋给 dr * /
dr. Read( ) ;//读取结果集中的行
node1. data. Enabled = ( bool) dr[ " qx" ] ;
/ * 取操作权限,并设置菜单项的 Enabled 属
性 * /
dr. Close( ) ;}//关闭数据读取器
con. Close( ) ;
```

至此，实现菜单深度优先遍历的算法设计完毕，下面结合实例介绍该算法的在具体的 MIS 系统中的应用。

5　在实际系统中的应用

使用 C#开发 MIS 系统时，只需根据上面的做法在项目中定义类 Cnode、QueueCnode 和 menusearch，并在 MIS 数据库中创建 tabmenu、tabuser 和 tabqx 三个表，就可以很方便地使用上面的算法遍历系统菜单，实现用户权限的动态控制。具体做法是，①用待遍历的系统菜单和数据库连接字符串作为参数创建一个 menusearch 类对象；②调用方法 Search 遍历菜单；③调用方法 WriteData 将菜单的遍历结果写入数据库；④调用方法 Setqx 读取用户权限，并设置相应菜单项的 Enabled 属性，实现权限控制。例如，某图书管理系统的主窗体是 FormMain，该窗体的主菜单是 menuStrip1，则要自动遍历该菜单，根据数据库中用户的操作权限来设置相应菜单项的 Enabled 属性，实现用户权限的动态控制，只需要为 FormMain 的 Load 事件编写如下代码：

```
menusearch smenu = new menusearch( menuS-
trip1, constr) ;/ * 创建菜单搜索类对象,参数 menuS-
trip1 是要遍历的系统菜单,constr 是存放系统数据库连
接参数的字符串 * /
smenu. Search( ) ;//自动遍历菜单
if( isfirst = = true) //如果系统是第一次运行
smenu. WriteData( ) ;/ * 把遍历结果写入数据库,
以便进行用户权限分配 * /
else
smenu. Setqx( user) ;/ * 读取用户权限,设置菜单
项的 Enabled 属性,从而控制权限 * /
```

其中，isfirst 用于标识系统是否是第一次运行，通常存放在系统配置文件中。若系统是第一次运行，则把菜单的遍历结果写入数据库，以便进行用户权限的分配；否则根据用户的操作权限设置系统菜单项的 Enabled 属性，实现用户权限的控制。

6　结束语

本文针对 MIS 系统中如何对用户权限进行动态控制，采用面向对象的方法，使用 C#实现了基于 C#的 MIS 系统菜单的深度优先遍历及系统权限的动态控制，该方法通用性和实用性极佳，有推广应用价值。

参考文献

1　张奇、李律松、卫建伟等，Visual C#数据库项目案例导航[M]，北京：清华大学出版社，2005.

2　张建英、王秀坤，一种面向应用的基于访问权限的动态控制方法[J]，计算机工程与应用，2003(07)：120.

3　熊小敏、周燕玲，PowerBuilder 下实现菜单级别的权限控制[J]，计算机与现代化，2003(12)：90.