【数据结构】Day4(2)

Class	Advanced Data Structures
Date	
Material	
Series Number	
Summary	

【Ch4】树

4.1 预备知识

在树的递归定义中我们发现，一棵树是N个结点和N-1条边的集合。

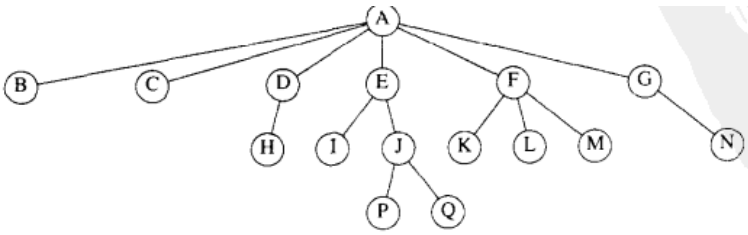

图 4-2　一棵具体的树

每条边都将某个结点连接到它的父亲，而除去根节点外每个结点都有一个父亲

对任意结点n_i, n_i的深度(depth)为从根到n_i的唯一路径的长。因此，根的深度为0。

n_i的高(height)是从n_i到一片树叶的最长路径的长，因此所有的树叶的高都是0

例：对于上图，E的深度为1（A为E的双亲结点），而高度为2（到最长树叶路径长为2）

4.1.1 树的实现

树的结点的所有儿子都放在树结点的链表中：

```
typedef struct TreeNode *PtrToNode;
struct TreeNode {
  ElemType Element;
  PtrToNode FirstChild; //第一个孩子结点
  PtrToNode NextSibling; //同层结点
}
```

4.1.2 树的遍历及应用

树的一个常见应用便是UNIX，DOS这些系统中的目录结构：

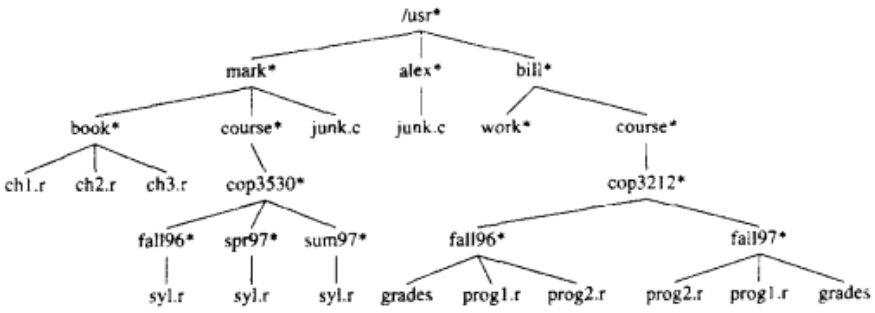

图 4-5　UNIX 目录

我们若想要遍历目录，打印每一份文件及文件夹得名字，函数如下：

```
        static void
        ListDir( DirectoryOrFile D, int Depth )
        {
/* 1*/      if( D is a legitimate entry )
            {
/* 2*/          PrintName( D, Depth );
/* 3*/          if( D is a directory )
/* 4*/              for each child, C, of D
/* 5*/                  ListDir( C, Depth + 1 );
            }
        }

        void
        ListDirectory( DirectoryOrFile D )
        {
            ListDir( D, 0 );
        }
```

图 4-6　列出分级文件系统中目录的例程

效果：

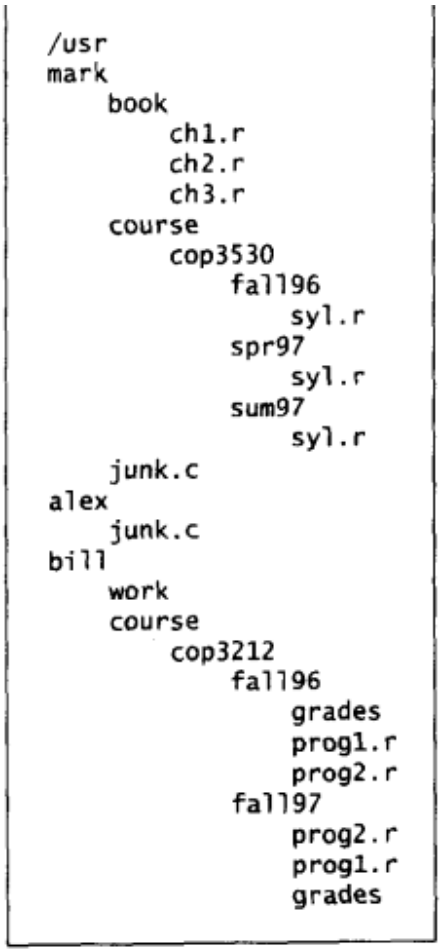

图 4-7　目录(先序)列表

遍历整个目录获得所有文件的大小：

```
            static void
            SizeDirectory( DirectoryOrFile D )
            {
                int TotalSize;

    /* 1*/      TotalSize = 0;
    /* 2*/      if( D is a legitimate entry )
                {
    /* 3*/          TotalSize = FileSize( D );
    /* 4*/          if( D is a directory )
    /* 5*/              for each child, C, of D
    /* 6*/                  TotalSize += SizeDirectory( C );
                }
    /* 7*/      return TotalSize;
            }
```

图 4-9　计算一个目录大小的例程

效果：

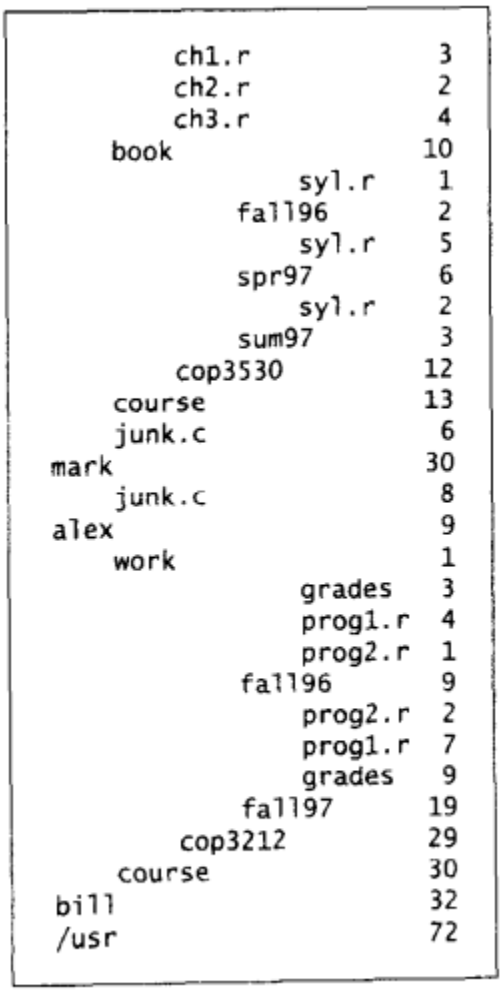

图 4-10　函数 SizeDirectory 的轨迹