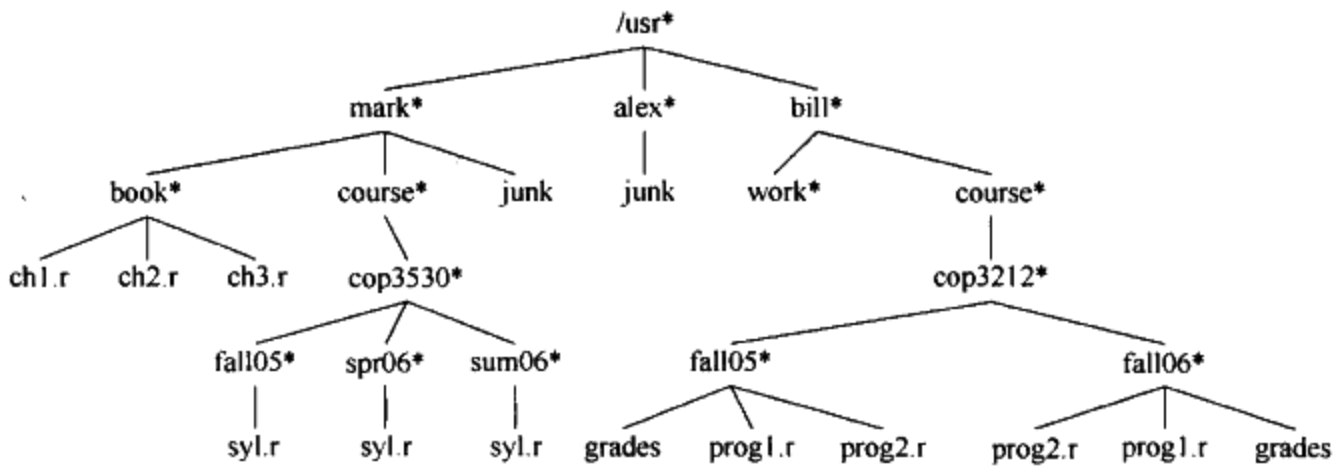

图 4-5 UNIX 目录

同的路径从而具有不同的路径名。在 UNIX 文件系统中的目录就是含有它的所有儿子的一个文件，因此，这些目录几乎是完全按照上述的类型声明构造的[㊀]。事实上，按照 UNIX 的某些版本，如果将打印一个文件的标准命令应用到一个目录上，那么在该目录中的这些文件名能够在(与其他非 ASCII 信息一起的)输出中被看到。

设我们想要列出目录中所有文件的名字。输出格式将是：深度为 d_i 的文件将被 d_i 次跳格(tab)缩进后打印其名。该算法在图 4-6 中以伪码给出[㊁]。

```
      private void listAll( int depth )
      {
1         printName( depth ); // Print the name of the object
2         if( isDirectory( ) )
3             for each file c in this directory (for each child)
4                 c.listAll( depth + 1 );
      }
      public void listAll( )
      {
          listAll( 0 );
      }
```

图 4-6 列出分级文件系统中目录的伪码例程

算法的核心为递归方法 listAll。为了显示根时不进行缩进，该例程需要从深度 0 开始。这里的深度是一个内部簿记变量，而不是主调例程能够期望知道的参数。因此，驱动例程用于将递归例程和外界连接起来。

算法逻辑简单易懂。文件对象的名字和适当的跳格次数一起打印出来。如果是一个目录，那么以递归方式一个一个地处理它所有的儿子。这些儿子均处在下一层的深度上，因此需要缩进一个附加的空间。整个输出在图 4-7 中表示。

这种遍历策略叫做**先序遍历**(preorder traversal)。在先序遍历中，对节点的处理工作是在它的诸儿子节点被处理之前(pre)进行的。很显然，当该程序运行时，第 1 行对每个节点恰好执行一次，因为每个名字只输出一次。由于第 1 行对每个节点最多执行一次，因此第 2 行也必然对每个

㊀ 在 UNIX 文件系统中每个目录还有一项指向该目录本身以及另一项指向该目录的父目录。因此，从技术上说，UNIX 文件系统不是树，而是类树。

㊁ 实现该算法的 Java 程序由文件 FileSystem.java 联机提供。它用到课文中尚未讨论的一些 Java 特点。