**MapInterface:**

**作用:**

1.DTN2发至MapInterface一个经纬度，判断这个经纬度是哪个区域

2.发送一个bundle时，需要根据一个区域id，判断它的父区域，DTN2发送至MapInterface一个区域id，判断它的父区域

**文件：**

sumo.db，由程序osm2sqlite生成。该文件按树形存储各个区域的情况。存储每个区域组成边的情况，子区域的情况和中心结点的情况。

**结构**：

1.Point.h只记录一个 点的经纬度。

2.Node.h 记录组成某个路的其中一个结点情况，包括结点的id,经纬度，结点在路的位置。

3.Way.h 记录组成某个边界的其中一个路的情况。包括组成该路的结点序列，路的id，该路在区域中的属性，是outer，还是inner。

4.Line.h记录组成某个区域的其中一个边界的情况。包括所属区域的id，组成这个边界的路序列，该边界的属性，是inner，还是outer。

5. AreaLayerInfo.h 记录经纬度所属区域的序列。

6. PointInPolygon.h，利用引射线法，判断经纬度是否属于某个区域（该区域由结点序列表示）

7. MapSqllite.h，打开sumo.db文件，对其执行两个查询。一个是查询某个区域的子区域情况，并回溯的判断某个经纬度是否属于该区域，如果属于，再次判断属于哪个子区域。一个是查询组成某个区域路径，组成路径的结点情况。再将这些信息转化为组成区域的结点序列，再根据节点序列判断经纬度是否处于该区域。

8. GetAdminAreaService.h，

1.接收客户端的请求。该请求由客户端ip，和经纬度组成。处理该请求，获得该经纬度所处的区域序列，将客户端ip，所属区域的个数，各个区域号转发给客户。

2.接受客户端的请求。该请求只由一个int型的区域id组成。处理后，获得区域的序列，并将区域的个数，底层区域id到顶层区域id序列发回客户端。

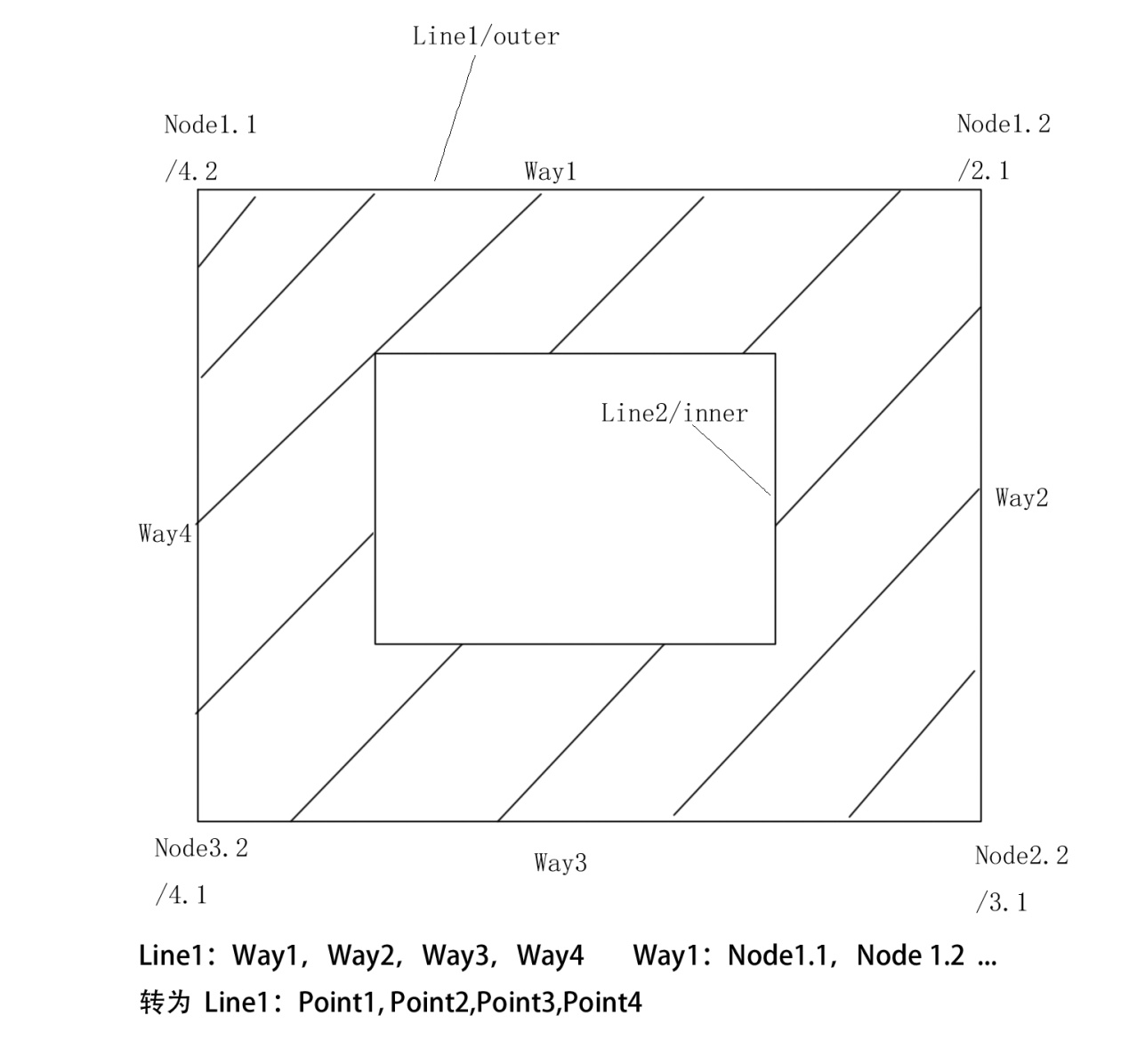
**流程：**

1.接受请求包，分解出经纬度和客户端ip，判断是否有该ip的记录，如果有，则判断该经纬度是否属于上次记录的区域，如果属于，说明区域没有转换，直接返回上次记录的区域信息。

如果不属于，或是没有该ip的记录，则从最上层的区域开始，从sumo.db文件中查询最上层区域的子区域情况，再判断经纬度属于第二层区域中的哪一个。按回溯的方法，依次判断，直到到最底层区域。

判断的方式为：首先查询该区域的路和结点情况，并一一处理存储至类line，way，node。即line代表区域的一个边界，way代表组成line的其中一条路，node代表组成way的一个结点（node是重复存储的，因为一个结点可以是多条路的交叉点），并将node按照在Way中位置按升序排序。以line中的最后一个way中的最后一个结点为开始结点，循环遍历每一个way，判断上一个way的最后一个结点是否是下一个way的开始结点。由此判断该line是否是一个封闭的区域，并得到组成line的结点信息序列。

最后利用该结点序列，判断经纬度是否是属于该区域。



2.启动程序后，该程序读取数据库，生成一个颠倒的树结构的区域层次树。当接收到请求后，根据这个树结构得到区域id序列，返回

**编译：**

编译时，加上 -lpthread,-lsqlite3;

在eclipse,则是在properties->C/C++ Build->Settings界面中

GCC C++ Linker->Libraries界面中，在Libraries(-l)中添加 pthread,sqlite3;

在makefile中 在LIBS := 加上 pthread sqlite3 即可

然后直接make