LumaQQ 开发者文档

LumaQQ

开源传万世 因有我参与

纯真IP数据库格式详解

摘要
网络上的IP数据库以纯真版的最为流行，LumaQQ也采用了纯真版IP数据库做为IP查询功能的基础。不过关于其格式的文档却非常之少，后来终于在网上找到了一份文档，得以了解其内幕，不过那份文档寥寥数语，也是颇为耐心才读明白。在这里我重写一份，以此做为LumaQQ开发者文档的一部分，我想还是必要的。本文详细介绍了纯真IP数据库的格式，并且给出了一些Demo以供参考。

Luma, 清华大学
修改日期： 2005/01/14

Note: 在此感谢纯真IP数据库作者金狐和那唯一一份文档的作者。

修改历史:
2005-01-14 修改了原来一些表达不清和错误的地方

自从有了IP数据库这种东西，QQ外挂的显示IP功能也随之而生，本人见识颇窄，是否还有其他应用不得而知，不过，IP数据库确实是个不错的东西。如今网络上最流行的IP数据库我想应该是纯真版的（说错了也不要扁我），迄今为止其IP记录条数已经接近30000，对于有些IP甚至能精确到楼层，不亦快哉。2004年4、5月间，正逢LumaQQ破土动工，为了加上这个人人都喜欢，但是好像人人都不知道为什么喜欢的显IP功能，我也采用了纯真版IP数据库，它的优点是记录多，查询速度快，它只用一个文件QQWry.dat就包含了所有记录，方便嵌入到其他程序中，也方便升级。

基本结构

QQWry.dat文件在结构上分为3块：文件头，记录区，索引区。一般我们要查找IP时，先在索引区查找记录偏移，然后再到记录区读出信息。由于记录区的记录是不定长的，所以直接在记录区中搜索是不可能的。由于记录数比较多，如果我们遍历索引区也会是有点慢的，一般来说，我们可以用二分查找法搜索索引区，其速度比遍历索引区快若干数量级。图1是QQWry.dat的文件结构图。

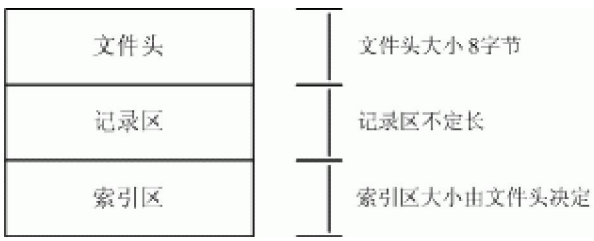

图1. QQWry.dat文件结构

要注意的是，QQWry.dat里面全部采用了little-endian字节序

一. 了解文件头

QQWry.dat的文件头只有8个字节，其结构非常简单，首四个字节是第一条索引的绝对偏移，后四个字节是最后一条索引的绝对偏移。

二. 了解记录区

每条IP记录都由国家和地区名组成，国家地区在这里并不是太确切，因为可能会查出来“清华大学计算机系”之类的，这里清华大学就成了国家名了，所以这个国家地区名和IP数据库制作的时候有关系。所以记录的格式有点像QName，有一个全局部分和局部部分组成，我们这里还是沿用国家名和地区名的说法。

于是我们想象着一条记录的格式应该是: [IP地址][国家名][地区名]，当然，这个没有什么问题，但是这只是最简单的情况。很显然，国家名和地区名可能会有很多的重复，如果每条记录都保存一个完整的名称拷贝是非常不理想的，所以我们就需要重定向以节省空间。所以为了得到一个国家名或者地区名，我们就有了两个可能：第一就是直接的字符串表示的国家名，第二就是一个4字节的结构，第一个字节表明了重定向的模式，后面3个字节是国家名或者地区名的实际偏移位置。对于国家名来说，情况还可能更复杂些，因为这样的重定向最多可能有两次。

那么什么是重定向模式？根据上面所说，一条记录的格式是[IP地址][国家记录][地区记录]，如果国家记录是重定向的话，那么地区记录是有可能没有的，于是就有了两种情况，我管他叫做模式1和模式2。我们对这些格式的情况举图说明：

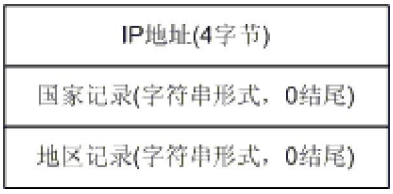

图2. IP记录的最简单形式

图2表示了最简单的IP记录格式，我想没有什么可以解释的

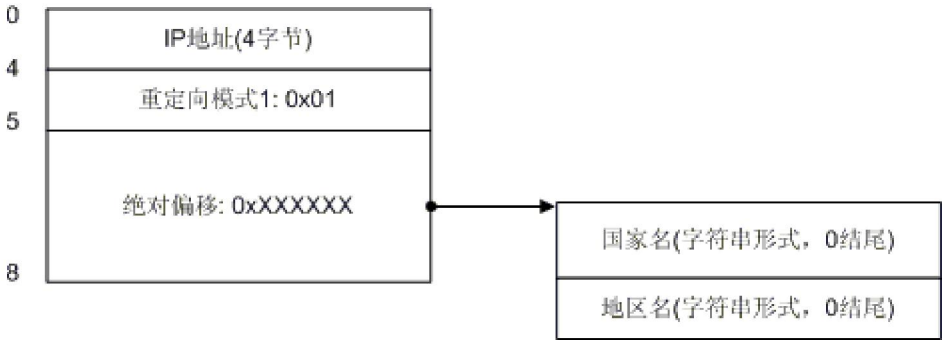

图3. 重定向模式1

图3演示了重定向模式1的情况。我们看到在模式1的情况下，地区记录也跟着国家记录走了，在IP地址之后只剩下了国家记录的4字节，后面3个字节构成了一个指针，指向了实际的国家名，然后又跟着地址名。模式1的标识字节是0x01。

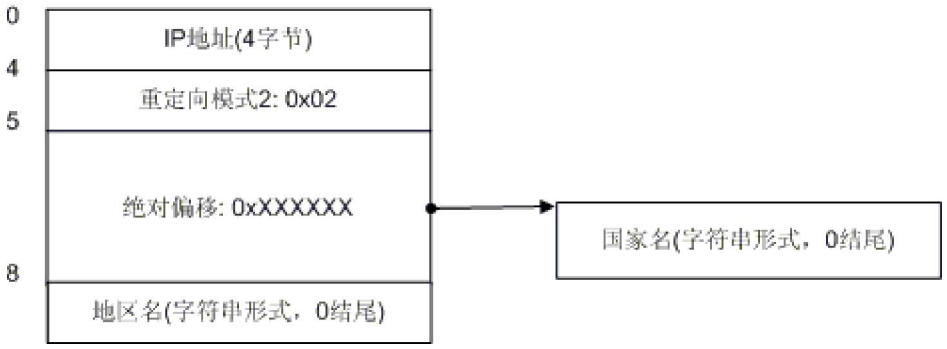

图4. 重定向模式2

图4演示了重定向模式2的情况。我们看到了在模式2的情况下（其标识字节是0x02），地区记录没有跟着国家记录走，因此在国家记录之后4个字节之后还是有地区记录。我想你已经明白了模式1和模式2的区别，即：模式1的国家记录后面不会再有地区记录，模式2的国家记录后会有地区记录。下面我们来看一下更复杂的情况。

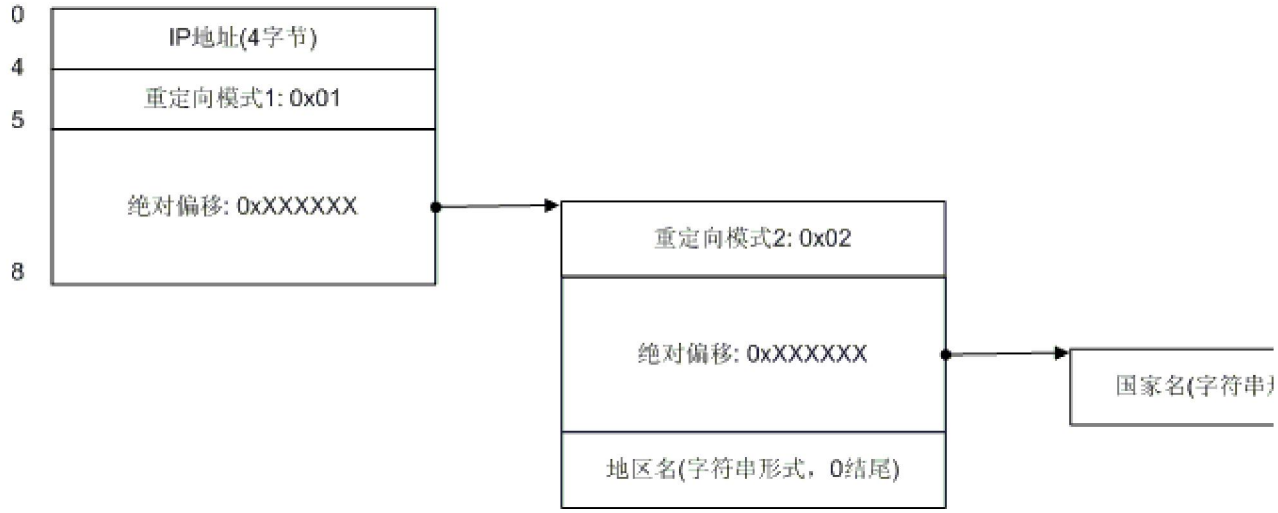

图5. 混和情况1

图5演示了当国家记录为模式1的时候可能出现的更复杂情况，在这种情况下，重定向指向的位置仍然是个重定向，不过第二次重定向为模式2。大家不用担心，没有模式3了，这个重定向也最多只有两次，并且如果发生了第二次重定向，则其一定为模式2，而且这种情况只会发生在国家记录上，对于地区记录，模式1和模式2是一样的，地区记录也不会发生2次重定向。不过，这个图还可以更复杂，如图7：

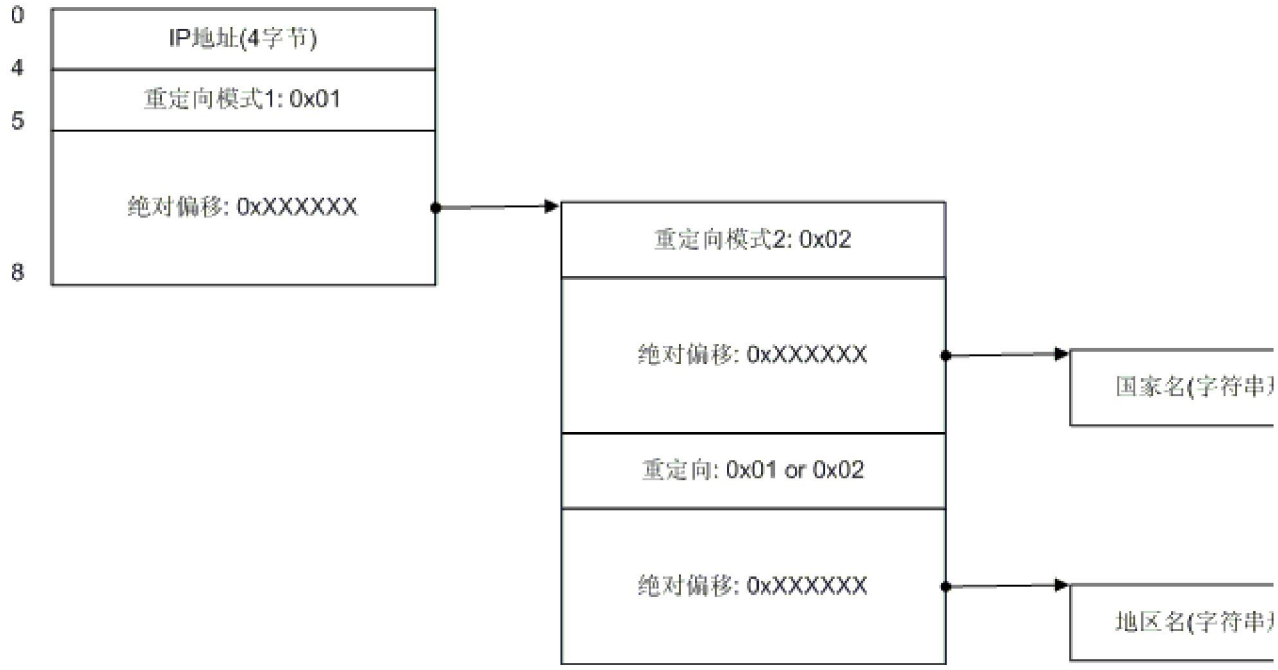

图6. 混和情况2

图6是模式1下最复杂的混和情况，不过我想应该也很好理解，只不过地区记录也来重定向而已，有一点我要提醒你，如果重定向的地址是0，则表示未知的地区名。

所以我们总结如下：一条IP记录由[IP地址][国家记录][地区记录]组成，对于国家记录，可以有三种表示方式：字符串形式，重定向模式1和重定向模式2。对于地区记录，可以有两种表示方式：字符串形式和重定向，另外有一条规则：重定向模式1的国家记录后不能跟地区记录。按照这个总结，在这些方式中合理组合，就构成了IP记录的所有可能情况。

设计的理由

在我们继续去了解索引区的结构之前，我们先来了解一下为何记录区的结构要如此设计。我想你可能想到了答案：字符串重用。没错，在这种结构下，对于一个国家名和地区名，我只需要保存其一次就可以了。我们举例说明，为了表示方便，我们用小写字母代表IP记录，C表示国家名，A表示地区名：

1. 有两条记录a(C1, A1), b(C2, A2)，如果C1 = C2, A1 = A2，那么我们就可以使用图3显示的结构来实现重用
2. 有三条记录a(C1, A1), b(C2, A2), c(C3, A3)，如果C1 = C2, A2 = A3，现在我们想存储记录b，那么我们可以用图6的结构来实现重用
3. 有两条记录a(C1, A1), b(C2, A2)，如果C1 = C2，现在我们想存储记录b，那么我们可以采用模式2表示C2，用字符串表示A2

你可以举出更多的情况，你也会发现在这种结构下，不同的字符串只需要存储一次。

了解索引区

在"了解文件头"部分，我们说明了文件头实际上是两个指针，分别指向了第一条索引和最后一条索引的绝对偏移。如图8所示：

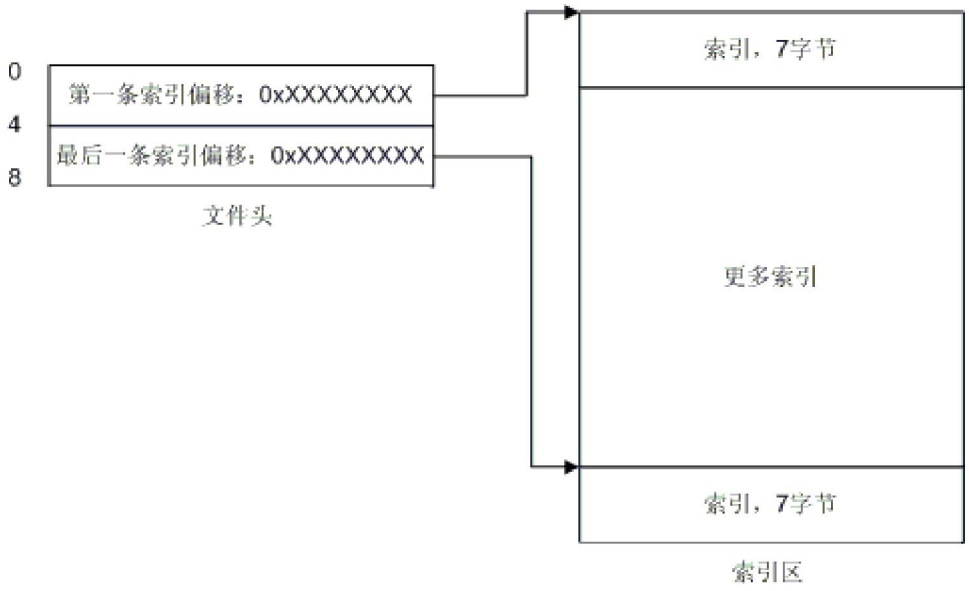

图8. 文件头指向索引区图示

实在是很简单，不是吗？从文件头你就可以定位到索引区，然后你就可以开始搜索IP了！每条索引长度为7个字节，前4个字节是起始IP地址，后三个字节就指向了IP记录。这里有些概念需要说明一下，什么是起始IP，那么有没有结束IP？ 假设有这么一条记录：166.111.0.0 - 166.111.255.255，那么166.111.0.0就是起始IP，166.111.255.255就是结束IP，结束IP就是IP记录中的那头4个字节，这下你应该就清楚了吧。于是乎，每条索引配合一条记录，构成了一个IP范围，如果你要查找166.111.138.138所在的位置，你就会发现166.111.138.138落在了166.111.0.0 - 166.111.255.255 这个范围内，那么你就可以顺着这条索引去读取国家和地区名了。那么我们给出一个最详细的图解吧：

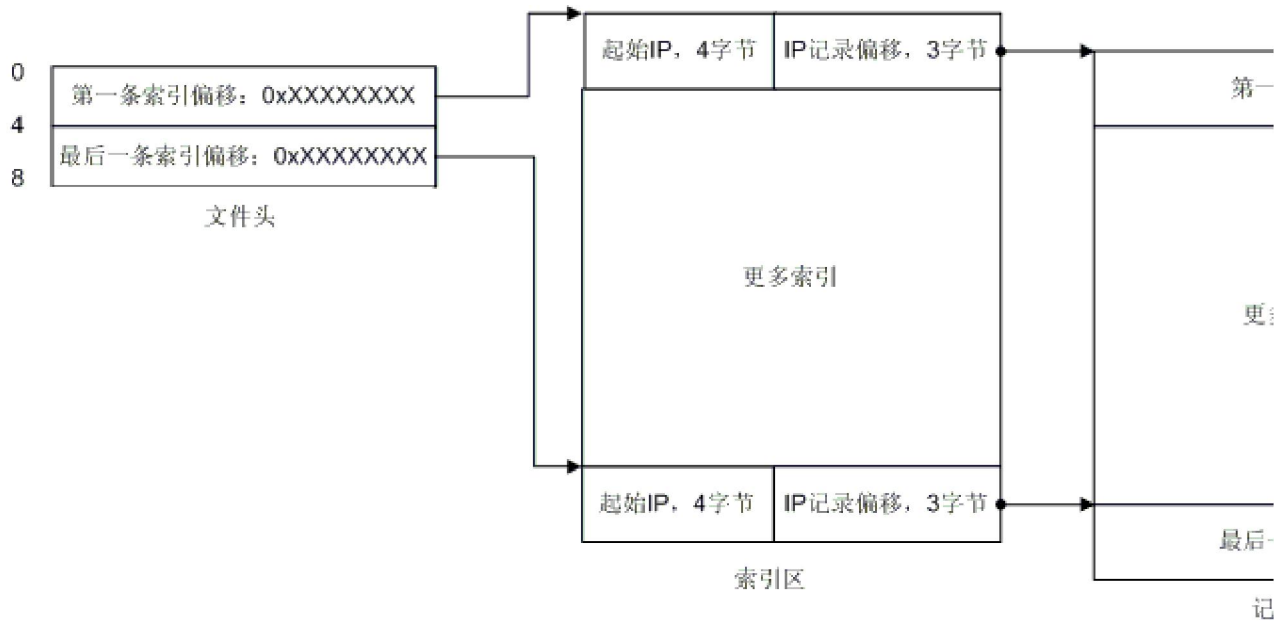

图9. 文件详细结构

现在一切都清楚了是不是？也许还有一点你不清楚，QQWry.dat的版本信息存在哪里呢？ 答案是：最后一条IP记录实际上就是版本信息，最后一条记录显示出来就是这样：255.255.255.0 255.255.255.255 纯真网络2004年6月25日IP数据。OK，到现在你应该全部清楚了。

Demo

下一步：我给出一个读取IP记录的程序片断，此片断摘录自LumaQQ源文件edu.tsinghua.lumaqq.IPSeeker.java，如果你有兴趣，可以下载源代码详细看看。

```
/**
 * 给定一个ip国家地区记录的偏移，返回一个IPLocation结构
 * @param offset 国家记录的起始偏移
 * @return IPLocation对象
 */
private IPLocation getIPLocation(long offset) {
	try {
		// 跳过4字节ip
		ipFile.seek(offset + 4);
		// 读取第一个字节判断是否标志字节
		byte b = ipFile.readByte();
		if(b == REDIRECT_MODE_1) {
			// 读取国家偏移
			long countryOffset = readLong3();
			// 跳转至偏移处
			ipFile.seek(countryOffset);
			// 再检查一次标志字节，因为这个时候这个地方仍然可能是个重
			b = ipFile.readByte();
			if(b == REDIRECT_MODE_2) {
				loc.country = readString(readLong3());
				ipFile.seek(countryOffset + 4);
			} else
				loc.country = readString(countryOffset);
			// 读取地区标志
			loc.area = readArea(ipFile.getFilePointer());
		} else if(b == REDIRECT_MODE_2) {
			loc.country = readString(readLong3());
			loc.area = readArea(offset + 8);
		} else {
			loc.country = readString(ipFile.getFilePointer() - 1);
			loc.area = readArea(ipFile.getFilePointer());
		}
		return loc;
	} catch (IOException e) {
		return null;
	}
}

/**
 * 从offset偏移开始解析后面的字节，读出一个地区名
 * @param offset 地区记录的起始偏移
 * @return 地区名字符串
 * @throws IOException 地区名字符串
 */
private String readArea(long offset) throws IOException {
	ipFile.seek(offset);
	byte b = ipFile.readByte();
	if(b == REDIRECT_MODE_1 || b == REDIRECT_MODE_2) {
		long areaOffset = readLong3(offset + 1);
		if(areaOffset == 0)
			return LumaQQ.getString("unknown.area");
		else
			return readString(areaOffset);
	} else
		return readString(offset);
}

/**
 * 从offset位置读取3个字节为一个long，因为java为big-endian格式，所以没办法
```

```
     * 用了这么一个函数来做转换
     * @param offset 整数的起始偏移
     * @return 读取的long值，返回-1表示读取文件失败
     */
    private long readLong3(long offset) {
              long ret = 0;
              try {
                        ipFile.seek(offset);
                        ipFile.readFully(b3);
                        ret |= (b3[0] & 0xFF);
                        ret |= ((b3[1] << 8) & 0xFF00);
                        ret |= ((b3[2] << 16) & 0xFF0000);
                        return ret;
              } catch (IOException e) {
                        return -1;
              }
    }

    /**
     * 从当前位置读取3个字节转换成long
     * @return 读取的long值，返回-1表示读取文件失败
     */
    private long readLong3() {
              long ret = 0;
              try {
                        ipFile.readFully(b3);
                        ret |= (b3[0] & 0xFF);
                        ret |= ((b3[1] << 8) & 0xFF00);
                        ret |= ((b3[2] << 16) & 0xFF0000);
                        return ret;
              } catch (IOException e) {
                        return -1;
              }
    }

    /**
     * 从offset偏移处读取一个以0结束的字符串
     * @param offset 字符串起始偏移
     * @return 读取的字符串，出错返回空字符串
     */
    private String readString(long offset) {
              try {
                        ipFile.seek(offset);
                        int i;
                        for(i = 0, buf[i] = ipFile.readByte(); buf[i] != 0; buf[++i] = ipFi
                        if(i != 0)
                            return Utils.getString(buf, 0, i, "GBK");
              } catch (IOException e) {
                  log.error(e.getMessage());
              }
              return "";
    }
```

代码并不复杂，getIPLocation是主要方法，它检查国家记录格式，并针对字符串形式，模式1，模式2采用不同的代码，readArea则相对简单，因为只有字符串和重定向两种情况需要处理。

总结

纯真IP数据库的结构使得查找IP简单迅速，不过你想要编辑它却是比较麻烦的，我想应该需要专门的工具来生成QQWry.dat文件，由于其文件格式的限制，你要直接添加IP记录就不容易了。不过，能查到IP已经很开心了，希望纯真记录越来越多～。

LumaQQ is a Java QQ client which has a reusable pure Java core and SWT-based GUI