从 Unicode 到 GBK 的内码转换

张晓培 李 祥

(贵州大学计算机软件与理论研究所 贵阳 550025)

摘 要:讨论了 GB,GBK,Big5,Unicode 编码以及内码转换原理和方法,介绍了在手机的 TTS 系统下把 Unicode 码转换成 GBK 码的方法,并使用代码页 CP936 中 Unicode 码和 GBK 码的对应关系实现了从 Unicode 码到 GBK 码的内码转换。

关键词: CP936 GBK Unicode TTS 内码转换

ISN Diversion From Unicode to GBK

ZHANG Xiaopei, LI Xiang

(Institute of Computer Science, Guizhou University, Guiyang, 550025, China)

Abstract: In this paper, we dessribe GB, GBK, Big 5, unicode code and the prirciple and method of code conversion. Then we introduce the method of conversion from wricode Fiually, we realize the convertion from unidce to GBK by using the relation of unicode and GBK code in code page 936.

Keywords: CP936, GBK, unicode, TTS, code comversion

1 汉字内码基本概念

1.1 GB 编码

GB2312 - 80《信息交换用汉字编码字符集基本集》,1980 年发布,是中文信息处理的国家标准,在大陆及海外使用简体中文的地区(如新加坡等)是强制使用的唯一中文编码。P - Windows3.2 和苹果 OS 就是以 GB2312 为基本汉字编码,Windows 95/98 则以 GBK 为基本汉字编码、但兼容支持 GB2312。GB 码共收录 6763 个简体汉字、682 个符号,其中汉字部分:一级字 3755,以拼音排序,二级字 3008,以偏旁排序。该标准的制定和应用为规范、推动中文信息化进程起了很大作用。1990 年又制定了繁体字的编码标准 GB12345 - 90《信息交换用汉字编码字符集第一辅助集》,目的在于规范必须使用繁体字的各种场合,以及古籍整理等。该标准共收录 6866 个汉字(比 GB2312 多 103 个字,其他厂商的字库大多不包括这些字),纯繁体的字大概有 2200 余个。(2312 集与 12345 集不是相交的。一个是简体,一个是繁体)。

1.2 Big5 编码

目前台湾、香港地区普遍使用的一种繁体汉字的编码标准,包括 440 个符号,一级汉字 5401 个、二级汉字 7652 个,共计 13060 个汉字。Big - 5 是一个双字节编码方案,其第一字节的值在 16 进制的 A0～FE 之间,第二字节在 40～7E 和 A1～FE 之间。因此,其第一字节的最高位是 1,第二字节的最高位则可能是 1,也可能是 0。

1.3 GBK 编码

GBK 编码(俗称大字符集)是中国大陆制订的、等同于 Unicode 的新的中文编码扩展国家标准。GBK 工作小组于 1995 年 10 月,同年 12 月完成 GBK 规范。该编码标准兼容 GB2312,共收录汉字 21003 个、符号 883 个,并提供 1894 个造字码位,简、繁体字融于一库。Windows95/98 简体中文版的字库表层编码就采用的是 GBK,通过 GBK 与 Unicode 之间一一对应的码表与底层字库联系。其第一字节的值在 16 进制的 81～FE 之间,第二字节在 40～FE,除去 xx7F 一线。

1.4 Unicode 编码

国际标准组织于 1984 年 4 月成立 ISO/IEC JTC1/SC2/WG2 工作组,针对各国文字、符号进行统一性编码。1991 年美国跨国公司成立 Unicode Consortium,并于 1991 年 10 月与 WG2 达成协议,采用同一编码字集。目前 Unicode 是采用 16 位编码体系,其字符集内容与 ISO10646 的 BMP(Basic Multilingual Plane)相同。Unicode 于 1992 年 6 月通过 DIS(Draf International Standard),目前版本 V2.0 于 1996 公布,内容包含符号 6811 个,汉字 20902 个,韩文拼音 11172 个,造字区 6400 个,保留 20249 个,共计 65534 个。

本文于 2004 - 04 - 26 收到。

2 手机 TTS 系统

2.1 手机 TTS 系统简介

随着移动通讯领域的飞速发展,越来越多的技术被应用到手机上来。本文所介绍的 TTS 是指在手机系统中,可以把一段文本文字转换成语音波形并使用普通话自动读出文本的系统。本系统主要进行文本分析,语音合成以及调用声卡硬件进行发音。首先系统中需要存入两种数据 —有关文字的字典库和有关发音的语音库。通过字典库可以找到相应的文字,根据一定的语法分析,词法分析以及语义分析并使用语音库进行合成语音的波形,从而实现文本变语音的功能。

2.2 和 Window 自带 TTS 功能的区别

在 Window 系统中,自带有微软制作的 TTS 功能,可以很方便的进行文本转换语音的工作。但是在手机系统中,却要受很多的限制:存放数据库不能占用太大空间,速度太慢会影响手机的性能等等。所以在手机中 TTS 功能的实现一定要进行数据存储格式的优化以及算法的优化。

3 内码转换原理及方法

3.1 内码转换原因

由于历史、地区原因,有时一种文字会出现多种编码方案,特别是汉字。由于不同于系统内码的字符不能在该系统中正常显示,必须要进行字符的内码转换,即将非系统内码的字符转换为系统可以识别的内码字符。在本系统中,TTS 中存储的中文字库是 GBK 编码的,所以对于 Unicode 码他并不能正确识别,所以必须在进行文本分析前进行从 Unicode 码到 GBK 码的转换。

3.2 内码转换原理及方法

内码转换就是在不同字符集之间建立一种对应关系。

对于每一种内码,都有一种自己的存储格式。在代码页 CP936 中,GBK 码是顺序排列的,而和 GBK 码一一对应的 Unicode 码却是不规则的。把 Unicode 码转换成 GBK 码,就要对 Unicode 码进行规则存储,而让 GBK 码和 Unicode 码进行一一对应,只要根据存储的规则找到了 Unicode 码,就可以通过映射找到对应的 GBK 码,从而完成内码的转换。

4 从 Unicode 到 GBK 的内码转换

4.1 Unicode 存储格式

在代码页 CP936 中,GBK 码的汉字有 20000 多个,相应的 Unicode 码的汉字也是 20000 多个,如果不进行转换直接存储,占用内存应该为 80K 左右。在 Windows 系统下 80K 可以说是九牛一毛,但是在手机的嵌入式系统中 80K 的花销却很大。

首先通过程序把 CP936 中的 Unicode 码和 GBK 码分别取出,并把 Unicode 码按照从小到大的顺序进行排列,GBK 码进行一一对应,并把 GBK 码存到一个大数组中。因为对于 ASCII 码,GBK 和 Unicode 码都是一样的,可自动识别而不需要转换,所以把这些可自动识别的内码去掉。

我们给 Unicode 码建立索引。每一个汉字的内码占两个字节,很多汉字内码的高位字节是一样的,只是低位字节不同而已,而 Unicode 码又是从小到大顺序存储的,所以如果我们使用它的高位字节作索引,再通过偏移量找到内码的位置,进行计算就可以得到它再数组中的位置,从而通过映射关系找到对应的 GBK 内码。存储关系图见图 1。

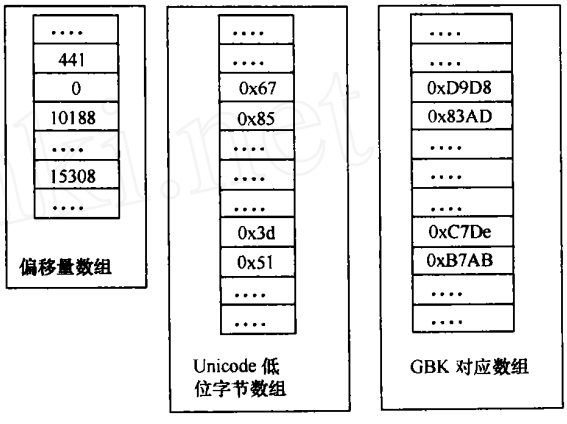

图 1 Unicode 和 GBK 对应关系和存储格式图

图 1 中,把 Unicode 码高低位分开,高位从 0x0n～0xff 但并不是完全连续,可能中间有些高位不存在。我们把以每一个高位开头的第一个 Unicode 码的位置作为偏移量存入偏移量数组中,如果高位不存在的我们填充为 0,这样数组的大小为 0xff + 1. 低位数组和 GBK 对应数组大小为 21791。

其中,GBK 数组占用 40K 左右的空间,Unicode 低位字节数组占用大约 20K 的空间,而偏移量数组只占用几十 bit 的空间,相当于节省了 20K 的内存空间。这对于收集的嵌入式系统来说是一笔很可观的价值。

4.2 转换具体算法

根据以上的存储格式,Unicode 低位字节的数组关系和 GBK 对应数组的关系是一一对应的。这样我们只要根据索引偏移量找到 Unicode 码低位字节在数组中的位置,那么在 GBK 对应数组中相应位置上的内码就是我们转换后所要得到的 GBK 内码。对于一个 Unicode 码的转换:

首先进行判断是否为 ASCII 码,如果是,则不需要转换直接使用,判断代码如下:

```
short is_ASCIWord(unsigned short unicode)
{if((unicode == (unsigned short)0x20AC)
||(unicode <= (unsigned short)0x007F
```

```
 && unicode >= (unsigned short)0x0000 ))
     return 1;
else
     return 0;}
```

如果不是 ASCII 码,则把它分成高位字节和低位字节两部分,我们把高位字节作为在偏移量数组中的位置下标量,找到该位置数组中存储的偏移量和下一个存储的不是 0 的数组中的偏移量,然后在 Unicode 码低位字节数组中进行顺序查找(因为低位字节数组中存储的格式是不规则的)。

```
static short binary_search
( short current_index,
short next_index,
unsigned short low_unicode)
{int i = 0;
for(i = current_index;i < next_index;i ++)
{if(low_Unicode = = unicode_low_data[i])
{return i;}}}
```

以上函数的返回值即为 Unicode 低位字节在 Unicode 码低位字节数组中的位置,由于 Unicode 码低位字节数组和 GBK 数组的关系是一一对应的,所以在 GBK 数组中这个位置上的 GBK 内码就是转换后所要得到的内码。

4.3　总结与思考

从存储结构和算法效率两方面的角度考虑,我们还可以使用哈希函数进行内码查找。这样我们只需要 40K 的空间存储 GBK 码的数组,使用哈希函数可以一次定位到 GBK 码的位置即可。但是由于两种内码的存储格式没有特定的规律,所以如何定位哈希函数会很麻烦,以及哈希函数的不稳定性可能会招致效率降低。

对于本文所介绍的转换算法,由于 Unicode 码低位字节不是按照顺序排列的也没有任何规则的排列,所以使用了顺序查找的方法。根据 Unicode 码按照高位相同的分组原则,索引的个数为 256 个,Unicode 码的个数为 20000 多个,这样平均每一个分组为 80 个左右,也就是说按照顺序查找平均查找 80 次就可以找到所要的内码。

总之,由于内码关系的存储格式不同,使用不同的算法就会产生不同的结果。所以内码转换的关键在于建立内码一一对应关系的存储格式,一个好的存储格式不仅可以占用更小的内存空间,而且在这种存储格式基础上的查找算法效率也会更高。

参考文献

1　http://www.5xsoft.com/data/200108/3108144301_1.htm
2　http://www.5xsoft.com/data/200108/3108144301.htm
3　http://school.enet.com.cn/document/20001115/2000111511382501.shtml
4　http://www.5xsoft.com/data/200109/0608094201.htm
5　台湾中国文化大学印刷传播系. 林威宇. http://www.cgan.com/science/publish/desktop/pdfchinese1.htm
6　http://www.bjprint.net/book/01/nm.htm
7　http://www.clyrics.com/stonec/hanzi/chinese.html
8　http://hunter.gnuchina.org/project/xml/
9　http://www.haiyan.com/steelk/navigator/ref/gbk/gbindex2.htm
10　http://www.99net.net/study/prog/67105540.htm