计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(7):251−255 [doi: 10.15888/j.cnki.csa.007533]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向信息系统跨平台轻量应用的汉字编码转换程序①

葛光富

(中国电子科技集团公司第二十八研究所, 南京 210007)
通讯作者: 葛光富, E-mail: 397559400@qq.com

摘　要: 国标码如 GB18030 是我国的汉字编码国家标准, UTF-8 是国际字符编码, 在国际化的今天这些编码方式在中文信息环境中同时存在并有着不少的使用. 为兼容与已有系统的如文本、协议的汉字交互处理, 新开发的信息系统必然需要将上述形式编码的汉字进行编码转换. 本文介绍了常用的汉字编码, 并详细说明了一种面向信息系统轻量应用的汉字编码转换程序, 该程序支持跨操作系统平台复用.

关键词: 信息系统; 跨平台; 轻量; 汉字编码转换; 国标码

引用格式: 葛光富.面向信息系统跨平台轻量应用的汉字编码转换程序.计算机系统应用,2020,29(7):251–255. http://www.c-s-a.org.cn/1003-3254/7533.html

Chinese Character Coding Conversion Program for Cross-Platform Lightweight Application of Information System

GE Guang-Fu

(The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210007, China)

Abstract: National standard codes such as GB 18030 is the national standard of Chinese character coding in China, and UTF-8 is an international character encoding. In the internationalization, these coding methods exist simultaneously in Chinese information processing environment. In order to be compatible with the existing systems, such as document and protocol Chinese characters processing, the newly developed information system must convert the Chinese characters in the above form. In this study, the common Chinese character coding standards are introduced, and a Chinese character coding conversion program for lightweight applications of information system is described in detail, which supports the reuse of cross-operating system platforms.

Key words: information system; cross-platform; lightweight; chinese coding conversion; national standard codes

引言

在我国的信息系统工程应用中, 顺应国际化时代的发展, 各类新运用的工具和软件的汉字编码大多采用了全球共享通用的 Unicode 字符集, 该字符集目前能够涵盖世界上主要语言的符号和文字. 而在指挥控制、网络安全、公共交通等业务领域方面, 现役信息系统的汉字编码包括有 GB2312、GBK 以及 GB18030 等, 为兼容与已有系统的互译互操作, 故按照国际化要求新开发的信息系统在相当的一段时间内, 仍会面临着上述编码形式的汉字编码转换需求[1–4].

综合军民用市场, 有着大量的计算处理设备用于搭建各型信息系统, 但这些设备却部署着种类多样、复杂不一的操作系统运行环境. 尤其是嵌入式设备更为突出, 如车载终端的 VxWorks (美国风河)、ReWorks

① 收稿时间: 2019-12-29; 修改时间: 2020-01-22; 采用时间: 2020-02-11; csa 在线出版时间: 2020-07-03

(电科 32 所) 等, 便携终端的 Android(美国谷歌)、WinCE (美国微软)、AOS (深圳华为)、SyberOS (北京元心) 等. 因此, 为降低系统开发维护成本、提升行业企业效益, 用于支撑应用跨操作系统平台快速移植改造的汉字编码转换技术, 成为信息系统软件服务平台统筹规划中的必要考虑因素.

要进行汉字编码转换, 对于 Linux/类 Linux 系统上的软件, 利用 GNU (GNU is Not Unix, 指的是一个自由软件工程项目) 的 libconv 库即可实现. 该库支持包括世界主流语系在内的字符集区域标准与国家标准编码间的互转, 但是这对于计算处理资源受限的嵌入式终端而言, 资源占用就显得有点庞大, 且不利于移植改造应用到各类操作系统尤其是国产化操作系统. 因此研究一种支持信息系统内部以及信息系统间通用的轻量化、可适用、易维护的汉字编码转换方法, 是在跨操作系统平台应用实践中急需解决的问题.

本文利用 Windows7 记事本工具, 进一步研究[5−7]开发出一种面向信息系统跨平台轻量应用的汉字编码转换程序, 能够提供有效的汉字编码转换接口, 用于与已有信息系统的如文本、信息的汉字交互处理, 支撑新开发信息系统的快速构建开通.

1 常用汉字编码

汉字编码指的是为汉字设计的一种便于输入电子计算机的代码, 是解决汉字能够进入计算机的关键. 国标码, 全名国家标准代码, 是我国的常用汉字编码集, 目前主要有 GB2312、GBK、GB18030 三种. 另外, UTF-8 因能够与 ASCII 兼容而作为优先采用的国际字符编码, 也涵盖了汉字的编码.

1.1 GB2312 编码

GB2312-80 编码是我国第一个汉字编码国家标准, 共收录汉字 6763 个, 同时收录了 682 个非汉字全角字符. 它对收录的每个字符采用两个字节表示, 其编码范围为 0xA1A1 到 0xFEFE, 首字节在 0xA1 与 0xFE 之间, 尾字节在 0xA1 与 0xFE 之间. 其中 0xB0A1 到 0xF7FE 为汉字的编码范围, 0xA1A1 到 0xA9FE 为非汉字字符的编码范围, 其他为空白区.

1.2 GBK 编码[5]

GBK 全称《汉字内码扩展规范》, 是在 GB2312 标准基础上的内码扩展规范, 使用了双字节编码方案, 其编码范围从 0x8140 到 0xFEFE, 首字节在 0x81 与 0xFE 之间, 尾字节在 0x40 与 0xFE 之间且不为 0x7F, 总共 23 940 个码位, 收录了 21 003 个汉字, 完全兼容 GB2312-80 标准, 支持国际标准 ISO/IEC10646-1 和国家标准 GB13000-1 中的全部中日韩汉字, 并包含了 BIG5 编码中的所有汉字. GBK 编码空间组成如表 1 所示.

表 1 GBK 编码空间组成

分类	编码范围	说明
GBK2 区	0xB0A1~0xF7FE	GB2312 汉字区
GBK3 区	0x8140~0xA0FE	GB13000.1 扩充汉字区
GBK4 区	0xAA40~0xFEA0	
GBK1 区	0xA1A1~0xA9FE	GB2312 非汉字符号区
GBK5 区	0xA840~0xA9A0	GB13000.1 扩充非汉字区
用户自定义 1 区	0xAAA1~0xAFFE	—
用户自定义 2 区	0xF8A1~0xFEFE	—
用户自定义 3 区	0xA140~0xA7A0	—

1.3 UTF-8 编码[6]

UTF-8 是一种针对 Unicode[7]字符集的可变长度字符编码, 所有的字符均使用 1 到 6 个字节进行编码, 是一种前缀码. 在只包含 1 个字节的 UTF-8 编码中, 其最高位置 0, 其余的 7 个二进制位用来对字符进行编码; 在含 $n(1<n\leq 6)$ 个字节的 UTF-8 编码中, 其第一个字节的前 n 位置 1, 第 $n+1$ 位置 0, 后续字节的最高位均置 1, 次高位均置 0, 全部字节中剩余的二进制用来对字符进行编码. 一个字符的 UTF-8 编码中包含字节的个数取决于其 Unicode 编码所处的范围, 具体如表 2 所示, 该表中*表示字符编码的可用二进制位, 将 Unicode 编码的二进制位按由低到高的次序放入*表示的空位中即可得到其所对应的 UTF-8 编码.

表 2 UTF-8 与 Unicode 编码对照表

Unicode 编码	UTF-8 编码
0x0~0x7F	0*******
0x80~0x7FF	110***** 10******
0x800~0xFFFF	1110**** 10****** 10******
0x10000~0x1FFFFF	11110*** 10****** 10****** 10******
0x200000~0x3FFFF	111110** 10****** 10****** 10******
FF	10******
0x4000000~0x7FFF	1111110* 10****** 10****** 10******
FFFF	10****** 10******

1.4 GB18030 编码

GB18030-2000《信息交换用汉字编码字符集基本集的扩充》, 是我国计算机系统必须遵循的基础性标准之一. 它是 GBK 的取代版本, 在其基础上增加了

6530 个 CJK(中日韩) 统一汉字扩充 A 的汉字 (由 4 字节表示: 第 1 字节 0x81~0x82, 第 2 字节 0x30~0x39, 第 3 字节 0x81~0xFE, 第 4 字节 0x30~0x39, 共占用了 25 200 个码位).

2 汉字编码转换程序分析与设计

因 GB18030 能够前向兼容 GBK、GB2312, 基于这一特点, 适应性选取 GB18030 与 UTF-8 间的编码转换来轻量应用支撑信息系统的汉字编码互处理需求, 同时使用通用的 C/C++语言, 分析与设计出一种面向信息系统跨平台轻量应用的汉字编码转换程序.

2.1 GB18030 与 Unicode 编码映射表制作

要进行 GB18030 与 UTF-8 间的汉字编码转换, 就必须利用 GB18030 和 Unicode 间的汉字字符码值映射关系, 以及 Unicode 和 UTF-8 间的国际字符编码转换规则, 故首先要制作 GB18030 与 Unicode 间的汉字编码映射表.

GB18030 编码中, GBK 汉字字符由 2 个字节表示, CJK 统一汉字扩充 A 的汉字字符由 4 个字节表示; 与其对应的 Unicode 标准字符集中, GBK、CJK 统一汉字扩充 A 的汉字字符编码占用 2 个字节大小的空间. 两者之间的汉字字符编码存在一一映射的关系, 可以通过编制汉字字符编码相互映射的表来进行转换.

GB18030 与 Unicode 编码映射表在互联网上百度文库、CSDN 社区等处能够找到部分缺漏不全的字符编码对照, 但在软件程序中使用还是需要经过二次加工制作成为可识别的代码语言. 而本文在 Windows7 系统平台上, 利用记事本工具和编写转换过程应用程序来辅助生成代码语言的汉字编码映射表, 具体的实现方法为:

1) 根据 1.2 和 1.4 小节中 GBK、CJK 统一汉字扩充 A 的汉字编码空间组成说明, 编写执行 GB18030 编码全汉字生成程序, 将全部有效的编码汉字按照码值从低到高依次保存到一个文本文件 GB.txt 中;

2) 在使用记事本方式打开 GB.txt 文件后, 将打开的文件另存为 Unicode 编码的新文件 Uni.txt 中;

3) 编写执行 GB18030 与 Unicode 编码映射表生成程序, 对照 GB.txt 与 Uni.txt 文件, 以 C 语言可识别代码的数组形式生成 GB18030 与 Unicode 间的汉字编码映射表, 以支持两者编码的快速映射查找. 映射表生成算法设计如算法 1 和算法 2.

算法 1. GB18030 到 Unicode 汉字编码映射表生成算法

1) 创建 GB18030 到 Unicode 汉字编码映射表, 其中数组支持存放元素 65 536 个, 数组元素下标能够一一对应 GB18030 汉字字符码值, 数组元素数值为 2 个字节大小存储的 Unicode 汉字字符码值, 初始化映射表元素值为代表无效映射的 0xFFFF;
2) 从 Uni.txt 文件中依次取得两个字节组成汉字字符, 获取汉字字符的 Unicode 码值;
3) 从 GB.txt 文件中依次取得两个字节, 根据两个字节的码值范围判断是否 GBK 汉字字符, 不是则再取得两个字节组成 4 个字节的 CJK 统一汉字扩充 A 的汉字字符, 获取汉字字符的 GB18030 码值;
4) 计算汉字字符的 GB18030 码值在按照码值从低到高的 GB18030 编码范围中的排序序号 (如 GBK 汉字的尾字节占用 191 个码位, 故 0x8140 对应序号为 0, 0x8240 对应序号为 191, 依次类推, 直到 CJK 统一汉字扩充 A, 也按照同样规则计算), 该序号为 Unicode 码值的存放位置;
5) 将汉字字符的 Unicode 码值, 存放到汉字编码映射表中对应的位置;
6) 返回第 2) 步, 直到文件结束, 生成 GB18030 到 Unicode 汉字编码映射表完毕.

算法 2. Unicode 到 GB18030 汉字编码映射表生成算法

1) 创建 Unicode 到 GB18030 汉字编码映射表, 其中数组支持存放元素 65 536 个, 数组元素下标为 Unicode 汉字字符码值, 数组元素数值为 2 个字节大小存储并能够一一对应 GB18030 汉字字符码值, 初始化映射表元素值为代表无效映射的 0xFFFF;
2) 从 GB.txt 文件中依次取得两个字节, 根据两个字节的码值范围判断是否 GBK 汉字字符, 不是则再取得两个字节组成 4 个字节的 CJK 统一汉字扩充 A 的汉字字符, 获取汉字字符的 GB18030 码值;
3) 汉字字符是 CJK 统一汉字扩充 A 汉字, 则计算汉字字符的 GB18030 码值在按照码值从低到高的 GB18030 编码范围中的排序序号 (如 GBK 汉字总计占用 24 066 个码位, 结合 CJK 统一汉字扩充 A 汉字的尾字节占用 10 个码位, 故 0x81308130 对应序号为 24 066, 0x81308230 对应序号为 24 076, 依次类推), 该序号为 GB18030 码值存表值, GBK 字符的 GB18030 码值存表值直接使用原值;
4) 从 Uni.txt 文件中依次取得两个字节组成汉字字符, 根据该汉字字符的 Unicode 码值, 获取汉字字符的 Unicode 码值, 该码值为 GB18030 码值相对值的存放位置;
5) 将汉字的 GB18030 码值存表值, 存放到汉字编码映射表中对应的位置;
6) 返回第 2) 步, 直到文件结束, 生成 Unicode 到 GB18030 汉字编码映射表完毕.

GB18030 和 Unicode 间的汉字编码映射表 (如汉字“科”—0xC6BF/0x79D1) 代码语言示例如图 1 所示, 映射表总计 1024 行, 每行 64 个码值映射元素, 其中值为 0xFFFF 表明当前位置对应的 GB18030 编码为无效编码.

有了汉字编码映射表, 就可以结合 UTF-8 与 Unicode 的编码转换规则, 实现字符串 GB18030 与

UTF-8 编码形式间的转换, 具体的转换程序设计见后续小节中描述.

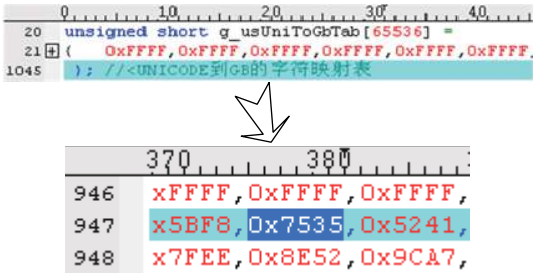

图 1 Unicode 到 GB18030 编码映射表

2.2 GB18030 与 Unicode 编码字符串转换

将 GB18030 编码字符串 (含英文、数字等基础 ASCII 码字符, 其码值范围为 0x0 到 0x7F) 转换到 UTF-8 编码的算法如算法 3.

算法 3. GB18030 到 UTF-8 编码字符串转换算法

1) 从 GB18030 编码字符串中依次读取一个字节, 根据码值判断是否基础 ASCII 码字符, 不是则再读取一个字节组成两个字节后判断是否 GBK 汉字字符, 不是则再读取两个字节组成 4 个字节的 CJK 统一汉字扩充 A 的汉字字符;
2) 字符类型是基础 ASCII 码字符, 则将字符按原码值作为 UTF-8 字符写入转换后的编码字符串, 否则转到下一步;
3) 根据汉字字符的码值, 计算字符在按照码值从低到高的 GB18030 编码范围中的排序序号, 查找 GB18030 到 Unicode 汉字编码映射表, 获取对应序号位置字符的 Unicode 码值;
4) 结合表 2 中体现的 UTF-8 与 Unicode 间的编码转换规则, 根据汉字字符的 Unicode 码值计算得到字符的 UTF-8 编码码值, 将相应码值的 UTF-8 字符写入转换后的编码字符串;
5) 返回第 1) 步, 直到待转换字符串结束, 生成转换后 UTF-8 编码字符串完毕.

GB18030 到 UTF-8 编码字符串转换利用接口函数 g2u() 实现, 关键代码如下:

```
ucChar = (unsigned char) pcInbuf[uiCharPos];
if (ucChar <= 0x7F) //<单字节字符, 直接替代
{
*(unsigned char *) (pcOutbuf + uiCharNewPos) = ucChar;
uiCharPos += 1;
uiCharNewPos += 1;
}
else //<非单字节字符, 查表映射到 Unicode
{
ucCharNext = (unsigned char) pcInbuf[uiCharPos+1];
if((ucChar >= 0x81) && (ucChar <= 0xFE)
&&(ucCharNext >= 0x40)
&&(ucCharNext <= 0xFE)
&&(ucCharNext != 0x7F)) //<GBK 双字节
{
puiWord = (unsigned short *) (pcInbuf + uiCharPos);
uiCharPos += 2;
}
else //<CJK 统一汉字扩充 A 四字节
{
puiWord = (unsigned int *) (pcInbuf + uiCharPos);
uiCharPos += 4;
}
//获取字符在 GB18030 编码范围的序号
iFindPos = GBCharSeqGet(puiWord);
//查表获取到 Unicode 码值, 并 Unicode 转到 UTF-8
uiCharNewLen =UnicodeToUtf8(g_usGbToUniTab[iFindPos],
(unsigned char *) (pcOutbuf + uiCharNewPos));
uiCharNewPos += uiCharNewLen;
}
```

将 UTF-8 编码字符串转换到 GB18030 编码的算法如算法 4.

算法 4. UTF-8 到 GB18030 编码字符串转换算法

1) 从 UTF-8 编码字符串中依次读取一个字节, 根据码值判断是否基础 ASCII 码字符, 不是则再读取一个字节直至到组成 UTF-8 字符;
2) 字符类型是基础 ASCII 码字符, 则将字符按原码值写入转换后的编码字符串, 否则转到下一步;
3) 结合 UTF-8 与 Unicode 间的编码转换规则, 根据 UTF-8 字符的码值计算得到 Unicode 码值, 查找 Unicode 到 GB18030 汉字编码映射表, 获取对应码值位置字符的 GB18030 码值存表值;
4) 根据 Unicode 码值的编码范围判断对应字符是否 CJK 统一汉字扩充 A 汉字, 是则 GB18030 码值存表值作为在 GB18030 编码范围中的排序序号计算还原出 GB18030 码值, 否则上一步存表值就是 GB18030 码值;
5) 将相应码值的 GB18030 字符写入转换后的编码字符串, 返回第 1) 步, 直到待转换字符串结束, 生成转换后 GB18030 编码字符串完毕.

UTF-8 到 GB18030 编码字符串转换利用接口函数 u2g() 实现, 关键代码如下:

```
ucChar = (unsigned char) pcInbuf [uiCharPos];
if (ucChar <= 0x7F) //<单字节字符, 直接替代
{
pcOutbuf [uiCharNewPos] = ucChar;
uiCharPos += 1;
```

```
        uiCharNewPos += 1;
        }
        else //<非单字节字符, 查表映射到 GB 字符
        {
        //UTF-8 字符先转到 Unicode
        uiCharNewLen = Utf8ToUnicode((unsigned char *)
(pcInbuf + uiCharPos), &usUniValue);
        //查表获取 GB18030 码值存表值
        usTemp = g_usUniToGbTab[usUniValue];
        bCJKaFlg = CJKaCharIfJundge(usUniValue);
        if(true == bCJKaFlg) //<CJK 统一汉字扩充 A 四字节
        {
        //计算还原 GB18030 码值
        uiTemp = GBCharValueGet(usTemp);
        pcOutbuf [uiCharNewPos] = uiTemp & 0xFF;
        pcOutbuf [uiCharNewPos + 1] = (uiTemp >> 8) & 0xFF;
        pcOutbuf [uiCharNewPos + 2] = (uiTemp >> 16) & 0xFF;
        pcOutbuf [uiCharNewPos + 3] = (uiTemp >> 24) & 0xFF;
        uiCharNewPos += 4;
        }
        else //<GBK 双字节
        {
        pcOutbuf [uiCharNewPos] = usTemp & 0xFF;
        pcOutbuf [uiCharNewPos + 1] = (usTemp >> 8) & 0xFF
        uiCharNewPos += 2;
        }
        uiCharPos += uiCharNewLen;
        }
```

3 程序应用实例

本文方法实现的汉字编码转换程序, 以动态库的形式提供, 适用于 GTK、Tilcon、Element-UI、QT 等界面库的信息系统软件开发, 并已成功运用于多型含装甲车载嵌入式、单兵移动便携信息处理终端的陆军业务信息系统中. 这些信息系统中, 配套工具开发的前端展现界面 (如图 2 所示) 的汉字编码类型大多数固定为 UTF-8 编码, 为兼容与已有系统如文本、协议的汉字交互处理, 后端服务处理采用的中文编码形式为 GB18030 或 GBK. 上述汉字编码转换程序不仅可以更轻量地应用于各类业务信息系统, 而且利于信息系统间的软件部件甚至整件的快速改造移植. 这些都能够为提高信息系统的资源利用率以及降低系统的开发维护成本, 起着积极的作用.

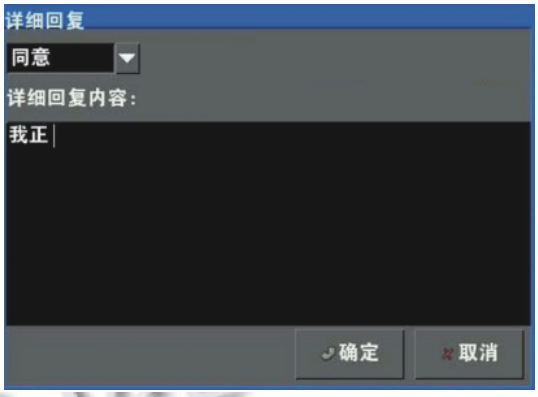

图 2 某装甲车载平台 GTK 开发的前端界面示例

4 结束语

本文立足于为信息系统的处理终端尤其是嵌入式终端提供轻量化的跨平台通用汉字编码转换手段, 论述了常用汉字编码的基本原理、编码对照关系. 同时, 给出了面向跨平台轻量应用的编码转换方案, 这套方案已成功地在多型嵌入式如单兵业务信息系统中得到应用[8], 且也能适用于其他信息系统[9].

参考文献

1 姚传明, 王庆元, 谢瑞生. 面向坦克的多目标威胁评估方法. 指挥信息系统与技术, 2018, 9(1): 68–72.
2 姚传明, 王庆元, 杨叶林. 基于 Tilcon 的多目标信息排序系统人机交互软件设计. 指挥控制与仿真, 2017, 39(3): 106–110. [doi: 10.3969/j.issn.1673-3819.2017.03.023]
3 王德泉. VxWorks 下汉字显示解决方案. 指挥信息系统与技术, 2010, 1(2): 61–65. [doi: 10.3969/j.issn.1674-909X.2010.02.014]
4 李元民. 将 MYSQL 的 GBK 数据库转成 UTF-8 数据库的简便方法. 广西民族大学学报 (自然科学版), 2006, (S1): 74–77.
5 杨善超. GBK 汉字编码技术研究. 福建电脑, 2017, 33(11): 13–30.
6 鹿文鹏, 薛若娟. Unicode 与 UTF-8 编码转换方法研究. 计算机时代, 2005, (9): 44–45. [doi: 10.3969/j.issn.1006-8228.2005.09.020]
7 张晓培, 李祥. 从 Unicode 到 GBK 的内码转换. 微计算机应用, 2006, 27(6): 757–759.
8 徐亮亮, 汤学达, 张媛, 等. 基于 AOS 的军用手持端即时通信软件设计. 指挥信息系统与技术, 2019, 10(5): 86–89, 100.
9 张庆海, 尹瑞, 代杰, 等. 面向网络化指挥控制系统的运维管理软件设计. 指挥信息系统与技术, 2018, 9(4): 68–73.