计算机科学 1995Vol. 22№. 3

中文信息处理国际化研究(上)

陈险峰*) 蔡希尧 金益民
(西安电子科技大学 西安 710071)

摘 要 The internationalization of Chinese processing is the one of the most important problems currently. Based on the "Universal Multiple Octet Coded Character Set"(ISO/IEC DIS 10646) standard and the MS-Windows 3.1 used as the platform, in this paper, the supporting approaches to ISO 10646 is studied, and the realizing method of input and output of UCS coded Chinese and English characters is given. Compatible with the Chinese Coding Standard GB 2312, the appropriate approaches to support the Chinese Coding System CJK defined in the UCS is also discussed.

关键词 Chinese processing, Information processing internationalization, ISO/IEC DIS 10646, UCS, CJK.

一、概 论

到目前为止,计算机代码处理体制是以传统的ASCII或者EBCDIC为基础的,采用八位编码,最多只能表示256个不同的字符,要处理所使用字符超过256个的非英语语种,特别是亚洲语系的语言,许多国家都制订了能表示本国文字的并与ASCII兼容的扩展编码字符集,如我国的GB2312-80、日本的JIS、台湾的TCA CNS等等。这些标准常常采用十六位或更多的位来表示一个字符,通过特定的转义序列进入或退出扩展字符集以达到对非ASCII字符的访问,再通过对软件系统的修改以达到对本国语言文字的处理。在我国最具代表性的工作就是西文软件的"汉化"。这种作法随之带来的问题有:

(1)软件系统的可移植性和一致性。虽然通过使用扩展字符集可以实现对非ASCII字符的处理,但是在以ASCII这样的八位字符代码体系为基础的软件系统中,一个扩展字符往往被当做一个ASCII字符串而非单个字符看待,也就是说,这些软件系统本身并没有从逻辑上支持扩展字符。加之不同国家、地区使用不同的字符编码,因而产生了歧义,特别是通过对目标码级程序的修改实现的"本地化"系统,这个问题就更加突出。

(2)增加了软件开发的难度和费用。一个软件系统为了适应不同的文化背景往往要开发出多个"本地化"版本,由于没有统一的字符编码,这就要求开发人员熟悉多种编码形式及多种文化背景,甚至一个软件系统为适应不同文化背景可能在不同"本地化"版本中将采用不同内核。如Microsoft公司的MS-Windows 3.1中文版系统就采用了双字节内核,而西文版系统是单字节内核。再如,由于中国大陆和台湾的汉字编码字符集不同,Microsoft公司分别推出了适用于中国大陆和台湾的两个MS-Windows 3.1中文版本,无疑造成了人力和物力的浪费。

(3)增加了系统运行时的开销。因为扩展字符集的使用,许多软件系统运行时不得不在不同字符集间来回切换。

因此,人们迫切希望有统一的对多文种处理的支持环境,能提供多文种接口,根据用户特定的国家、语言和文化习惯,建立适当的人机交互界面。汉语(汉字)作为世界上使用人数最多的语言(文字),对它的研究在国际化的研究中占有更为重要的地位。为了最终保证应用系统在中国与世界的可移植性,并使国际化的进程对我国信息产业的发展产生积极的意义,中文信息处理技术必须标准化并融入国际化的大潮流之中。

1984年,ISO开始研究一个新的字符编码标准。1992年5月,能表示世界上多种文字的"通用多八位编码字符集"ISO/IEC DIS 10646(简称UCS)获

*)现在电子部六所工作。

得通过，其目的是代替原有分散的、自成体系的编码字符集。把世界上各国常用的字符都收集在同一个编码字符集中，以超集的形式构成，以便适用于多文种信息的处理，减少指明调用字符集所带来的开销。UCS 是我国参与工作最多的国际标准，其中的一个重要内容是中日韩统一汉字，它的应用和推广，必将对中文信息处理产生重大影响。

我们的研究工作的目的是在现有操作系统上提供对多文种的处理支持，具体工作包括以下几个方面：

(1)研究在 MS-Windows 3.1 上支持 UCS 的途径和方法。

(2)在 MS-Windows 3.1 上实现通用多八位编码字符集编码的中英文字符的输入输出，寻求一种同时支持多文种的方法。

(3)在满足与现有的 GB2312 汉字编码标准相兼容的前提下，研究支持 CJK 汉字所要做的工作和方法。

我们把目前所完成的系统缩写为"XDUCS"。

二、ISO 10646 及其支持方法简介

1. 设计目标

原有的多文种代码标准体系"ISO 2022 信息交换用七位编码字符集的扩充方法"，主要适用于通信而不宜于信息处理，UCS 是在 ISO 2022 代码体系之内建立的一个独立的新体系。

UCS 的设计力求达到以下几个目标：为实现者提供一种平稳的向 UCS 迁徙的途径，通过对现存字符处理模块进行适当扩充就能方便地实现这种迁徙；使得使用本地语言的开发者更易于开发新产品；保证能与其它标准及其它系统交互运作；为世界多种文本中的每一个图形字符提供代码。在信息系统内，UCS 被设计成能用于世界上各种语言的书面形式以及附加符号的表示、传输、交换、处理、存储、输入及展现。

2. UCS 的总体结构

2.1 UCS 的代码形式

UCS 有两种代码形式，即由四个八位组成的正则形式 UCS-4 和两个八位的 UCS-2，下面分别介绍：

(1)UCS 的正则形式：用四个八位表示每个字符，并相应地指定组(group)、平面(plane)、行(row)和字位(cell)，分别简称为 G、P、R、C 八位。UCS-4 的编码空间大小为 128×256×256×256，其字符的代码结构如图 1 所示。

m. s. l. s.

Group-octet 组八位	Plane-octet 平面八位	Row-octet 行八位	Cell-octet 字位八位

图 1 UCS 字符正则形式代码结构

(2)UCS-2：也称位 BMP 形式，是 UCS 四维代码空间中的第一平面(00 组 00 平面)，也叫做基本多文种平面，可用作双八位编码字符集而单独使用，也称为 UCS-2。在基本多文种平面中包含字母文字、音节文字及表义文字中通常使用的字符以及各种符号与数字。此外，还包含一些限制使用区。目前的 UCS 版本只定义了基本多文种平面。UCS-2 字符的代码结构如图 2 所示。基本多文种平面分成四个区，其结构如图 3 所示。

m. s. l. s.

Row-octet 行八位	Cell-octet 字位八位

图 2 UCS-2 字符代码结构

	00 FF
00	A 区(19903 个图形字符位置)
4E	I 区(20992 个图形字符位置，已收入 20902 个汉字)
A0	O 区(16384 个图形字符位置)
E0	R 区(8190 个图形字符位置)
FF	

图 3 基本多文种平面结构

A(Alphabetic)区：用来安排字母文字、音节文字及各种符号。在本区的 00 行中安排的是 ASCII 字符，其中的 00 到 1F 的代码位置仍为控制字符。这样安排的好处可以做到使用 ASCII 的软件向使用 UCS 迁徙时，软件修改量最小。

I(Ideographs)区：用来安排中、日、韩统一的表义文字。

O(Open)区：未用。

R(Restricted)区：用于专用字符、变形显现字符(如阿拉伯文)和兼容字符。

2.2 其他规定

UCS 的 00 组中除基本多文种平面的后 255 个平面作为辅助平面，将容纳附加的图形字符。

UCS 中 60 到 7F 的 32 个组位专用组，00 组中从 E0 到 FF 的 32 个平面为专用平面。

用于表示图形字符的 P、R、C 各八位值必须具有从 00 到 FF 范围内的值，而组八位值必须具有从 00 到 7F 范围内的值。任何平面中，不应使用代码位置 FFFE 和 FFFF。

由于 UCS 字符组成的数据，不能直接经由对"控制字符"敏感的系统（如字符通道或面向字符的通信）传送，因此，在 ISO/IEC DIS 10646 中规定了一种 UCS 的变换格式（UTF-UCS Transformation Format）和同 UCS 的变换函数，供按 ISO 2022 结构对编码控制字符敏感的通信系统使用。

2.3 逻辑字符集概念和 UCS 的两类子集

UCS 与现存的字符集（如 ASCII 或 GB2312-80）标准系列不同，那些字符集总是把字符的代码及其物理实现紧密相连。而 UCS 是一种"逻辑"的字符集，它只为世界上的每一个字符分配一个确切的代码，以作为计算机系统中各组成部分的共同界面。对 UCS 中的每一个字符或者每一个已定义的汇集（collections）是否做物理上的实现，以及如何做物理上的实现则全由实现者自己决定。

UCS 引入了两种图形字符子集的概念，以便由源设备及接收设备进行交换，它们分别是：

2.3.1 有限子集 一个有限子集由指定子集中的图形字符的清单组成。这种子集规格使得使用了其它代码（如 GB2312）而开发的应用系统及设备能与本编码字符集交互运作。这种交互运作通常要涉及两种代码之间的转换，它为现存系统向使用 UCS 平稳迁徙提供了一个途径。

2.3.2 选择子集 在 UCS 中定义了若干字符汇集，每一个汇集都有确定的汇集号、汇集名及汇集中字符的位置区域。一个选择子集就由 UCS 中已定义的图形字符的清单组成。选择子集总是自动包含 00 组 00 平面 00 行从 20 到 7E 的字位，即 ASCII 中的所有图形字符。选择子集为有选择地实现 UCS 中的各个汇集的组合提供了实现的灵活性。

2.4 UCS 中汉字子系统介绍

在 UCS 的基本多文种平面的 I 区定义了 20902 个中、日、韩统一汉字，也称为 CJK 汉字。这些汉字构成了 UCS 的一个汇集。UCS 中的 CJK 汉字收集了中国大陆（GB）、台湾（TCA-CNS）、日本（JIS）、韩国（KSC）现存字符集中的常用汉字，并且将这些汉字统一编码。编码的顺序主要以汉字笔划为序。

UCS 中将汉字同西文字符统一编码，汉字同西文字符具有同样的代码长度，被当作同样类型的数据处理。由于提供了一种定长的无歧义的字符，从而使得系统内部的实现变得更加简练。同时，由于 CJK 汉字是对中、日、韩所使用的常用汉字进行统一编码，为中文信息处理和国际化提供了一个统一的基础。

但是，CJK 汉字的编码顺序同我国目前的国家标准 GB2312-80 及其它扩展汉字编码标准的编码顺序不同，同台湾、日本、韩国的汉字编码标准的编排顺序也有差异。CJK 汉字同 GB2312 汉字之间没有函数关系实现两者之间的相互转换，这在不抛弃大量原有产品的情况下，给实现对 CJK 汉字的支持带来一定的困难。

三、XDUCS Windows 的设计

1. XDUCS Windows 的设计目的

应用系统国际化的一个重要环节是把为特殊文化服务的程序从应用系统中分离出来，将这些界面标准化并在软件平台中提供国际化服务。因此，应用系统的人机接口中对具体文化的支持最终是由软件平台完成的。也就是说，只有在提供国际化支持的软件平台上，应用系统的本地化版本才能为特定的文化环境服务。因此，提供国际化支持的软件平台是实现国际化的基础。

长期以来，我国在信息处理中主要使用汉字编码字符集 GB2312-80，在应用中已取得很多成果，也积累了大量有用的信息资源。因此，在 ISO 10646 的推行过程中，GB2312 在今后很长的一段时间中还将继续被广泛使用。出于兼容性的考虑，多种代码标准（包括 ASCII）将会同时存在，同时运作，这样，代码之间的相互转换是必不可少的。基于以上的考虑，对支持 UCS 的方法和要做的工作必须做细致的研究。

一般来说，一个采用全新的编码字符集的软件平台应该重新构造，或者在已有系统的源码上修改完成。但重新构造一个系统的工作量极大，修改系统源码的条件不成熟，缺乏足够的投资。在这种情况下，可行的办法是对现有平台的目标码进行适当修改来达到对 ISO 10646 的支持，特别是研究使用 UCS 实现对汉字的支持。但这种方法无法修改操作系统核心，因此不可能从根本上做到对 UCS 的全面支持，仅仅只限于一些局部范围，如字符的输入和输出。所以，我们目前工作的主要意义是为后继的工作探索道路，积累经验。

我们选择 MS-Windows 3.1 英文版做为目标平台，希望对系统进行最少的修改达到对 UCS 的支持，主要实现 UCS 字符的输入/输出、支持使用 UCS 数据的程序运行。选择 MS-Windows 3.1 的主要理由有以下几点：

(1)MS-Windows 3.1 是当前世界上最流行的操作环境之一，系统的规模适中，并提供了丰富的应

用程序接口(API)。

(2)通过在 MS-Windows 3.1 上的工作,可以为以类似的系统(如 X-Window)上的类似工作提供一些借鉴,也可以为在 Windows-NT 上扩充汉字支持打下一定基础。

2. XDUCS Windows 的设计思想

MS-Windows 3.1 英文版本身有一定的国际化支持能力,它采用单字节编码字符和内核,能支持以拉丁语系为主的几种西方语言。XDUCS Windows 的设计目标是试图通过使 MS-Windows 对 UCS 的支持实现对多文种的统一支持,特别是在同一个字符集上对西文和东亚表义文字的统一支持。我们目前的实现是对具有代表性和现实意义的英文和中文的支持,但我们仍按照对多文种统一处理的模式设计,只是把英文和中文作为两种具体语言看待。

按照 UCS 实现的要求,采用有限子集和选择子集的混合形式。其中,有限子集定义为由 GB2312 中所有的汉字字符在 UCS 的对应字符组成。选择子集由两个汇集组成,即汇集 1 IRV-646(位置 0020 到 002F,即 ASCII 中的所有图形字符)和汇集 58 CJK UNIFIED IDEOGRAPHS(位置 4E00 到 9FFF,即 CJK 统一汉字)。

由于 ISO/IEC DIS 10646 目前的版本中只定义基本多文种平面,在本系统中我们只实现对 UCS-2 的支持。

修改目标系统的实现方法,不能修改 Windows 的核心使之从根本上直接支持 UCS。通过对系统的扩充实现 UCS 字符的输入/输出方法大致有两类:

(1)以应用程序的形式开发系统。这种方法保留了原 Windows 的所有特性,利用 MS-Windows 3.1 SDK 构造支持应用程序使用的 UCS 字符输入/输出程序(包括汉字的输入/输出)。这种方法对 Windows 的修改最小,可以保证所有应用程序的兼容性,缺点是执行的效率会受一定影响,另外,在应用程序级上建立系统有可能同其他应用程序发生冲突,导致系统崩溃。

(2)通过修改 Windows 的输入/输出设备驱动程序来支持 UCS。通过修改键盘驱动程序可直接支持 UCS,修改显示器驱动程序可以显示 UCS 字符,特别是汉字。这种方法也可以保留 Windows 的全部特性,也能保证应用程序的兼容性。缺点是要在 MS-Windows 3.1 DDK 支持下开发,另外,对于每一个不同的键盘驱动程序和显示器驱动程序都要做类似的改动才能使 Windows 在各种环境下都支持 UCS 并保证应用程序的可移植性。这样做的工作量将非常大。

我们采用"并存式"模式,理由是:(1)由于无法修改 Windows 采用单字节核心和编码字符集核心,对于至少由两个字节组成的 UCS 字符,我们只能将它分解成两个连续的单字节字符在 Windows 中流动。所以说这样一种对 UCS 字符的支持只是一种"伪支持"。(2)在 XDUCS Windows 内部流动的数据可能采用 ASCII、GB2312 和 UCS,我们不可能使系统内部只支持唯一的 UCS 字符集,也无法准确划分出一个系统边界。

3. CJK 汉字同 GB2312 标准的相互转换关系

在 ISO 10646 的推行过程中,会在很长一段时间内和目前已有的字编码标准并存,CJK 汉字同 GB2312 标准间的相互转换是不可缺少的。

CJK 汉字收集了中国、日本、韩国的常用汉字 20902 个,汉字的排列主要是以汉字偏旁部首和笔划为序。GB2312 标准只是 CJK 汉字一个子集,它只收集了我国的常用汉字 6763 个,其中,汉字的排列次序同 CJK 汉字的排列次序也有很大不同。由于不同汉字的使用频率相差很大,GB2312 标准由一级字库和二级字库两个子集构成。其中,一级字库中的汉字的使用频率高,这个子集中的汉字是按汉字的拼音次序排列的;二级字库中的汉字使用频率明显低于一级字库中的汉字,该子集中的汉字是按汉字的偏旁部首和笔划排列的。因此,CJK 汉字同 GB2312 标准间没有一个简单、直接的函数关系可以实现两者间的相互转换。因此,构造转换表实现两者间的相互转换是一种简单易行的方法。

通过转换表实现转换,转换效率和转换表的组织将对系统的效率产生很大影响,下面我们分别讨论几种转换表的组织方式和对应的转换方式。在下面的讨论中,我们将所有汉字的编码(无论是 UCS 编码还是 GB2312 编码)都做为一个无符号整数看待,并认为在内存中一个无符号整数由两个字节表示。

(1)分别构造两个从 GB2312 标准到 CJK 汉字转换表 A 和从 CJK 汉字到 GB2312 标准的转换表 B,这两个转换表都可以用线性数组实现,数组元素为一个汉字转换的目标集合内的编码。两个转换表都按各自的转换源集合中的汉字编码次序排列。分别用一个汉字的两个编码为索引值查找对应的编码。这种方法不需要额外的计算和比较操作,具有最快的转换速度,但占用的内存空间大,约需要 54K

左右。

(2)只构造从 GB2312 标准到 CJK 汉字的转换表 A，这个转换表用线性数组实现，并可完成从一个汉字的 GB2312 编码到 UCS 编码的转换。从 CJK 汉字到 GB2312 编码的转换通过查找表 A 中的元素并计算该元素在表中的偏移来实现。当表 A 按 GB2312 标准的编码次序组织时，表 A 的内容(对应的汉字 UCS 编码)是无序的，可采用顺序查找方法。

假设每个汉字出现的频率相同，则成功查找的平均查找长度为：(1＋2＋…＋6763)/6763＝3382，不成功查找的查找长度为 6763。

这种方法将进行多次比较操作，转换速度最慢，但占用最少的内存空间，只需要 13K 左右。在内存要求苛刻的情况下可以考虑采用。

(3)做为方法一和方法二的折中，首先构造从 GB2312 标准到 CJK 汉字的转换表 A，这个转换表用线性数组实现，并可完成从一个汉字的 GB2312 编码到 UCS 编码的转换。然后构造一个表 B，表 B 的元素为一个指向表 A 的一个元素的指针，元素的组织次序可有多种方式：

①元素按所指向的表 A 元素的值(一个汉字的 UCS 编码)的从小到大次序(也可按相反次序)排列，从 CJK 汉字到 GB2312 编码的转换可通过对表 B 的元素进行一次间址后的有序表二分查找并计算在表 A 中的偏移得到。这种方式下一次成功查找的最大查找次数为：$[\log_2 N]=[\log_2 6768]=13$，不成功查找的最大查找次数为：$[\log_2 N]+1=[\log_2 6763]+1=14$。

②元素按所指向的表 A 元素的使用频率的高低次序排列，从 CJK 汉字到 GB2312 编码的转换可通过对表 B 的元素进行一次间址后的表顺序查找得到并计算在表 A 中的偏移得到。假设 GB2312 标准中按区位码次序的汉字的使用频率分别是 p1，p2，…，p6763，那么，成功查找的平均查找长度为：p1＋2×p2＋…＋6763×p6763，其中，p1|p2|…|p6763＝1。并且有 0<p6763≤…≤p2≤p1<1。不成功查找的查找长度为 6763。

这种方法将进行几次比较操作和一次间址操作，其转换速度低于方法一，但高于方法二；内存空间的占用量也居前两者之间，约需要 27K。

经我们统计，在这两个标准中，恰好具有相同排列次序的汉字数目不超过 9 个。因此，如果构造分段的转换表性能不一定能有很大改观。另外，还可以构造哈希函数实现转换。总之，采用什么方法实现两种标准间的转换应根据实际情况而定。

4. XDUCS Windows 的体系结构

我们以应用程序形式开发并运行，其体系结构如图 4 所示。

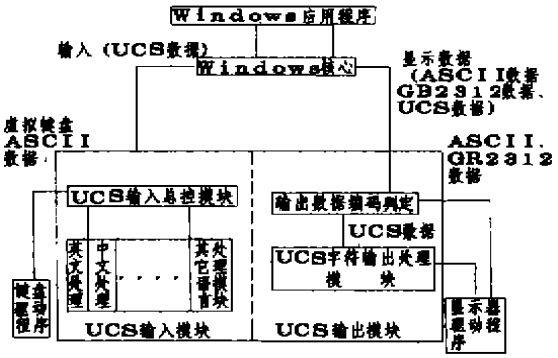

图 4　XDUCS Windows 体系结构

XDUCS Windows 的主要工作是支持 UCS 字符的输入和显示，因此，系统主要由输入、输出两个大模块组成。这两大模块位于 Windows 核心和设备驱动程序之间。输入模块根据当前用户选择的语言，将键盘输入或者键盘输入序列转换成为一个两字节的 UCS 编码，并形成两个字符消息发送给 Windows 或应用程序。显示输出模块首先判断所要显示数据采用的编码形式，对于采用 UCS 编码的数据，由 UCS 字符输出模块处理，对非 UCS 数据则调用 Windows 原来的字符输出函数将其输出。(未完待续)

(上接第 39 页)

参考文献

[1] Anoop Gupta, Parallelism in Production Systems, Morgan Kaufmann Publishers, Inc. California, 1987

[2] 孙钟秀，谢立，90 年代的分布式计算机技术，计算机学报，第 3 期，1992

[3] 祝旺发，分布式人工智能，计算机研究与发展，第 10 期，1990

[4] C. L. Forgy, Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence 19, Sept. 1982

[5] Steve Kuo and Dan Moldovan, The State of Art in Parallel Production Systems, Journal of Parallel and Distributed Computing 15, 1992