汉字字符问题简述

姜孟冯

这篇文章主要讨论如何在计算机上处理汉字字符信息的问题。

其实汉字字符问题分为三个部分，第一个问题就是输入，没有输入计算机不可能得到汉字信息。第二个问题是储存，汉字要想象普通的数据一样传输、交换，不把其以一定的格式储存下来是不行的。最后一个问题是输出，在显示器上显示出汉字，也要让打印机能打印出汉字。

一、输入问题

输入问题中最主要的显然是键盘输入问题，语音输入和手写识别之类虽然已能应用，但显然不及键盘输入普遍。这里不得不说一下的是汉字编码。潘德孚先生、詹振权先生的《汉字编码设计学》一书中说:“汉字编码是把汉字编为键盘上的符号代码。”而汉字编码与汉字键盘输入法的不同在于：“按照一定的规则，使汉字组成一套能拼、线性、有序性符号系统，称为汉字编码；而利用汉字编码、键盘设备、计算机资源和输入员将汉字输入计算机的方法称为汉字键盘输入法。”必须要弄清楚我们要解决的问题是什么，决不能混要视听，无的放矢。

这是一个什么样的问题呢？抽象出来看它的话，现在有两个集合，一个是汉字集合 A，一个是键盘输入的键位线性组合的集合 B，我们要做的就是在 B 与 A 之间找一个关系，使输入的键位线性组合对应一个或多个汉字。这里为什么可以对应多个汉字呢？实际上是为了方便考虑，汉字太多，每个汉字都规定一个键位线性组合，像电报码一样，又有谁能记得住呢。划分成等价类，分而治之，才是解决之道。

既然是键盘输入，就需要考虑键盘的因素。

首先说一下日式键盘，日式键盘把一个键对应到一个元音上，由于日文里所有单词（单語）都可用元音拼得，故只要输入这个单語的元音排列，就可以把这个单語输入进计算机。然而中文并没有像元音一样的东西，如果说勉强说有的说，只能算拼音或者注音了，但中文汉字是表意型文字，用发音来标定实在有很大的时空局限性，不说各地方言发音天差地别，光看现在网上流行的新词汇越来越多，很多字不会读不知道如何打出，就知道以发音标定汉字实在是不妥。例如“囧”这种网上聊天常用字，很少有人知道读 jiong3；又如日文中的姬与中文中的姬，在字符串检索中难道要因为 ji1 打不出前一个字而导致查询失败么。

目前中国普遍使用的是英式 101/104 键盘，与日式 106/109 键盘不同的是，并不存在可由固定一个键确定元音，每一个键只能对应一种新型规定的符号，然后将这个符号串对应到汉字上去。拼音、注音是解决途径之一，也可以有其他方法。早期的笔形码输入繁琐，现在恐怕已无人问津；形音码虽然自成流派，但如果是一字多部首、一部首多读音的情况都存在问题，唯一确定性强差人意。字根码以高频（组字频度高、使用频度高）字根作为编码方案的基本要素，其典型代表的五笔输入法以高速为主要竞争手段，但它的字根很多都是无理或半无理的，不死记硬背根本不行。潘德孚先生把汉字拆成部件，然后以字形来标定汉字，倒是有理有据，他虽然确定了一个用汉字的全部“块”标定汉字的标准，但知道的人不多，现在也没有流行起来。

电报码使用整个汉字；四角号码使用整个汉字的四角笔形；笔形码使用笔画；字根码使用汉字

的部分“块”；部件码则使用汉字全部的“块”。这些都是从字形方面出发来构建一个从键位线性组合到汉字集合的一个关系。我不清楚输入繁体字的人用什么输入法，但从简体字输入方面来看，上述那些编码在今天并不如字音编码流行。现在流行的微软拼音、智能 ABC、紫光、拼音加加、搜狗、谷歌、等等等等，哪一个不是以拼音为基础的输入法？

字音编码之所以能够流行，是跟时代脱不开关系的。近十年来，中国的家庭电脑普及率越来越高，同时国人的受教育程度也越来越高。网上聊天之类的已经成为了人们消遣的一种。打字，越来越进入人们的生活，并不需要什么打字比赛，也不需要什么极高速的输入法，打字已经融入人们的生活之中，易学易记，想打字的时候随手打出，这才是最好的。

很显然拼音正具备了这种特性，迅速的普及并流行，这是时代的必然产物。

看一下其他国家的输入法，日本流行的是 Microsoft IME Standard 2002 (ver8.1)，我个人是用日文 xp 的。正如前文所述一样，日文所有词都可以用元音拼的，假名直接输入，汉字按读音拼出，甚至一些键盘上没有的符号都可以用拼的，不像中文输入法必须调出表来查。例如，☆这样一个字符，我打ほし(hoshi)(意思为星)，然后在列表中选择即可。从我一个在韩留学的同学那里得知，韩国人常用的是 Korean Input System (IME2002)。由于韩文的文字是把表音的元音堆成方块成字，输入时也是按照读音来输入，因此也是字音编码。可惜韩文的键位很不对外国人口味，상(sang)这样一个字要打 tdk，在外国人看来相当的荒谬。这是因为，韩国人设计键盘时把它们的元音写在键盘上，ｔ ㅅ、ｋ ㅏ、ｄ ㅇ，打字时必须按照他所设计的键位来打，至于为什么这样设计，我不懂，想知道的去问韩国人吧……

为什么中国拼音比较流行，看看其他国家都是按音输入也可以大致想象了吧……

正如很多人虽然都喜欢用拼音但爱好的输入法软件各不相同一样，现在的输入法已经越来越“亲切”了，不但要求智能组句，而且其词库功能显得尤为重要，要会自我学习，还要便于修改，搜狗的词库搜罗了网上诸多流行词汇，谷歌可以让你把词库发去邮箱，日文 IME 可以实现词库成块成块的封装。各输入法生产厂商所注重的，基本都是在软件层次上一较高低。现在已经不是再去探索什么新的汉字编码的时代了，我认为，这是当前的一种文化。

“中国人要走进一个全民使用计算机的时代，也要与外国人一样，让计算机进入中小学课堂。计算机将进入基础教育；汉字编码，也将列入中小学生的识字教育课程之中。这就是说，书写工具的改变，增添了基础教育的内容，引发了识字教育的变革。说明我们已面临汉字应用史上为一次重大转折，每一个人都应该认真关注这件大事。汉字编码的好坏，不仅关系到社会使用的方便与否，而且关系到基础教育，这是中华民族子子孙孙的千秋大业。因此，我们需要进行艰苦的理论探索。”

这是潘德孚先生、詹振权先生的《汉字编码设计学》绪论中的一段话，看得出来，十几年前，计算机汉字编码深受邓爷爷的“计算机要从娃娃抓起”影响，跟教育联系在了一起。然而，事实证明，90 年代教育带给我们的拼音，已经成为当前流行的主要汉字编码手段。而当前的中小学教育，主要注重的是素质的提高，改革主要体现在英语上，至于计算机这东西，仅仅在操作应用的角度教一点点而已。

然而使用汉字的全部“块”的部件码，并不是一无所获的。它科学地论述了汉字编码的重大意义，使中国人认识到了排序的重要性，打破了万码奔腾的局面。汉字编码问题，实际上就是汉字排

序问题，也就是把汉字集映射到一个全序集上。 “键盘上的符号是有序的，因此，汉字编码就是为汉字设计一套有序的符号系统。”一旦排序确定了，什么都好办。

无论按照部首+笔画还是按照拼音，汉字都只能构成一个偏序集，而且问题多多，例如一字多音，例如有些难检字部首不好确定，甚至还有些汉字连一致认可的笔画数都没有。

这是一个标准的问题，而标准在计算机对汉字的储存上，明显有着比输入更为重要的意义。

二、储存问题

汉字的储存问题，其实就是一个标准的问题，要制定一个标准出来，让计算机能够把汉字唯一确定的以数据的方式保存。

抽象出来看，就是从汉字集合 A 的子集 C 到一个 01 串集合 D 的函数，每个汉字对应的 01 串必须唯一确定，这个 01 串称为汉字的内码。英文的内码就是 ASCII 码，汉字的内码呢，要靠国家制定标准了，这就是要要确定一个有序的汉字字符集。这个字符集上的汉字对应的内码唯一确定。

要制定这个汉字字符集，主要有两大困难：一是选字难，二是排序难。汉字包括简体字、繁体字、日本汉字、韩国汉字，数量庞大，而字符集空间有限，必须择字而用；排序难前文已经说明，概不赘述。

同时，在内码选定还需要注意：1.不能有二义性，即不能和 ASCII 码有相同的编码。2.要与汉字在字库中的位置有关系，以便于汉字的处理、查找。3.编码应尽量短。

从计算机的应用开始,我国已经颁布了多种中文信息编码标准,常用的是 GB2312-1980,GB13000（GBK），以及最新标准 GB18030。

GB2312-80

国家标准 GB 2312-80《信息交换用汉字编码字符集基本集》于 1980 年发布使用，它奠定了我国中文信息处理技术的发展。

1984 年“全国计算机与信息处理标准化技术委员会”提出编码字符集的繁体字和简体字对应编码的原则，并做出了制定六个信息交换用汉字编码字符集的计划。这六个集分别命名为基本集、第一辅助集(辅一)、第二辅助集(辅二)、第三辅助集(辅三)、第四辅助集(辅四)、第五辅助集(辅五)。其中，基本集、辅二集、辅四集是简体字集，辅一集、辅三集、辅五集分别是基本集、辅二集、辅四集的繁体字映射集,且简/繁字在两个字符集中同码(个别简/繁关系为一对多的汉字除外)。

基本集(GB 2312-80)，又称国际字符集。收入汉字信息交换用的基本图形字符，包括 3 部分组成：第一部分是字母、数字和各种符号，包括英文、俄文、日文假名、希腊字母、汉语拼音、注音字母等 687 个；第二部分为一级常用汉字，共 3755 个，按汉语拼音排列；第三部分为二级常用汉字，共 3008 个，按偏旁部首排列。总计简化汉字 6763 个，图形字符 7445 个。

它为其中任意一个字符(汉字或其他字符)规定了一个唯一的二进制代码。码表由 94 行(十进制编号 0-93 行)、94 列(十进制编号 0-93 列)组成，行号称为区号，列号称为位号。每一个汉字或符号由区号 7 位二进制代码与行号 7 位二进制代码唯一确定。这 14 位区位代码即为汉字的区位码。汉字的国标码又与之略有不同，由于信息传输的原因，每个汉字的区号和位号各加上 20h 才是该汉

字的国标码。实际在计算编码上，汉字内码采用双字节表示，每个字节的最高位置 1，以与 ASCII 码区别。汉字的区位码和国标码均是唯一的、标准的，而汉字内码会随系统不同有差别。

辅二集(GB 7589-87)和辅四集(GB 7590-87)是作为基本集的补充而编制的，均收通用规范的简体汉字，分别收字 7237 和 7039 个，都以部首为序排列，部首次序按笔画数排列，同部首字按部首以外的笔画数排列，同笔画数的字以笔形顺序(横、直、撇、点、折)为序。

辅一集(GB 12345-90)已于 1990 年发布，是与基本集对应的繁体字集，共收图形字符 7583 个，其中前 15 区除收集了 GB 2312 中前 15 区内收的全部字符外，又增收了 35 个竖排标点符号和汉语拼音符号。从 16 区至 91 区共收 6866 个繁体汉字。一级汉字数和二级汉字数都与 GB2312 相同，另有 103 个繁体字是属于简/繁为一对多的字。对于简/繁一对多的情况，则选一个最通用的繁体字码置于与基本集中该字相对应的码位，其余的则按拼音序编码于 88 和 89 区。

辅三集和辅五集分别是辅二集和辅四集的一一对应的繁体字符集，比辅二集和辅四集中的字有更多的使用机会。

这六个集在编码上大致相若，均采用双七位编码方式，每张代码表分 94 个区和 94 个位。其中前 15 区作为拼音文字及符号区或保留未用，16 区到 94 区为汉字区。

之后又有了第七辅助集，其汉字的来源是 GB13000.1 的 CJK 统一汉字部分，为日本、韩国和台湾地区使用的汉字。

GB13000(GBK)

GB13000(GBK)的到来是信息全球化进程的产物，为了适应多文种的同时处理，国际标准化组织下属编码字符集工作组研制了新的编码字符集标准，ISO/IEC 10646。该标准第一次颁布是在 1993 年，当时只颁布了其第一部分，即 ISO/IEC 10646.1: 1993，我国相应的国家标准是 GB 13000.1-93《信息技术 通用多八位编码字符集(UCS) 第一部分：体系结构与基本多文种平面》。制定这个标准的目的是对世界上的所有文字统一编码，以实现世界上所有文字在计算机上的统一处理。

GB13000 建立了一个全新的编码体系。ISO/IEC 10646 被称作"多八位"编码字符集，是因为它采用四个"八位"（即 8 bit 或称作字节）编码。这四个字节被用来分别表示组、平面、行和字位。

GB13000 的总编码位置高达 2,147,483,648 个（128 组×256 平面×256 行×256 字位）。目前实现的是 00 组的 00 平面，称为"基本多文种平面"（Basic Multilingual Plane, BMP），编码位置 65536 个。（由于基本多文种平面所有字符代码的前两个字节都是 0（00 组 00 平面 XX 行 XX 字位），因此，目前在默认情况下，基本多文种平面按照两字节处理。）

GB13000 的优点和特点非常明显：

编码空间非常巨大，可以容纳多种文字同时编码，也就保证了多文种同时处理；

作为统一的编码，拉丁语系的文字与其它文字一样，都是采用相同数目的"八位"编码，即：都是四字节，在基本多文种平面，都是双字节；注：对于 GB1988（ISO646/ASCII）字符，直接增加高八位为 0x00 即可。

字符和字形的区分十分清楚：字符是负载文本内容的抽象实体，而字形则是可视的具体图形形式；

通过采用汉字认同规则，各国家/地区的汉字统一编码，既满足了各国家/地区对编码汉字数目的实际需求，又不至于由于汉字在基本多文种平面占据的码位过多而影响到其他文字的编码：

由于世界上的文字数量巨大，不可能将所有文字编码，为此，划定了专用区，供标准使用者实现其对未编码字符的特别需要。

其码位分配简图（GB13000.1-200X 版）如下：

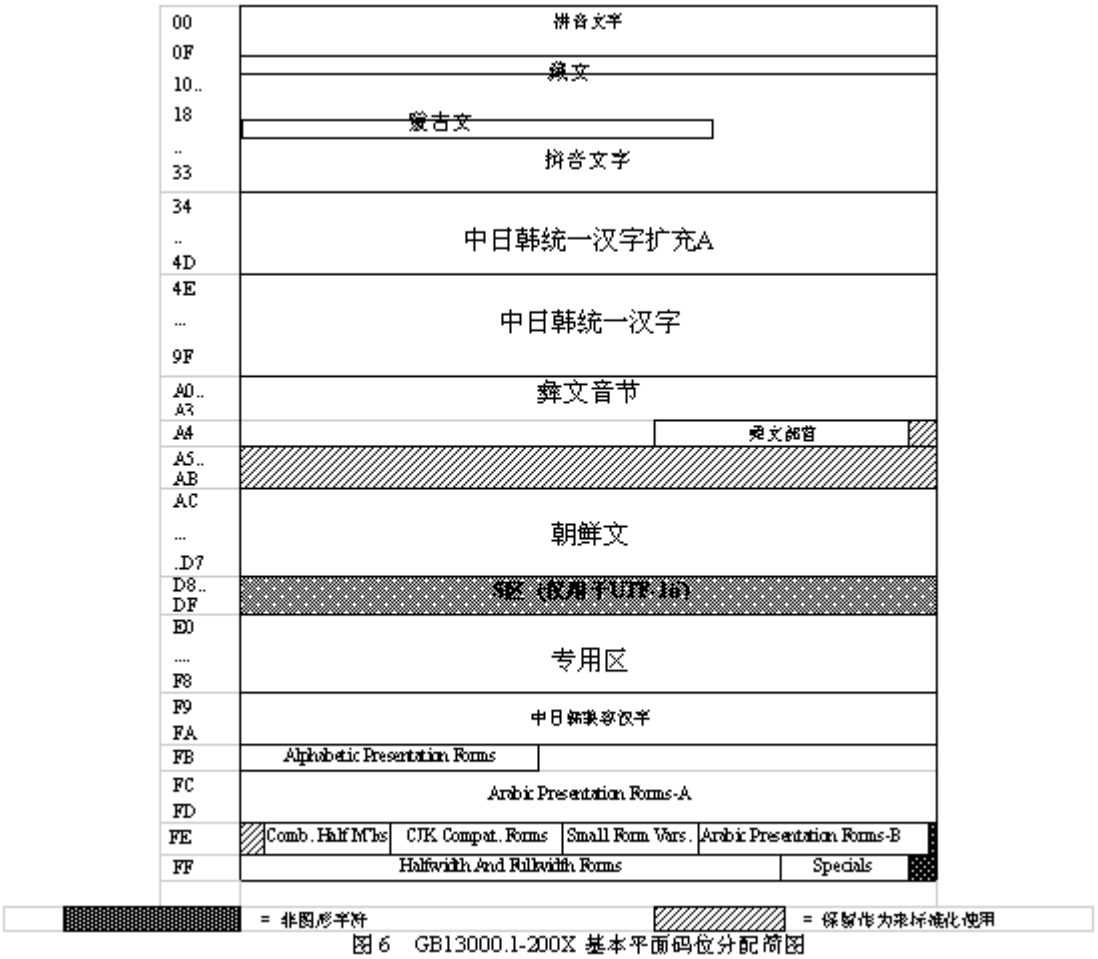

图 6　GB13000.1-200X 基本平面码位分配简图

其中，CJK 统一汉字和 CJK 统一汉字扩充 A 收录了 GB2312 和第一、三、五、七辅助集的全部汉字 27，484 个。康熙部首和中日韩补充部首共收录汉字部首 369 个。

GB18030

如果说 GB13000 是 UCS 在中国的体现，那么 GB18030 是受 unicode 影响的产物。GB18030 也是对 GB2312 的扩展，其编码长度由 2 个字节变为 1~4 个字节。其中包括：

* 单字节，其值从 0 到 0x7F。

* 双字节，第一个字节的值从 0x81 到 0xFE，第二个字节的值从 0x40 到 0xFE（不包括 0x7F）。

* 四字节，第一个字节的值从 0x81 到 0xFE，第二个字节的值从 0x30 到 0x39，第三个字节的值从 0x81 到 0xFE，第四个字节的值从 0x30 到 0x39。

可以看出，GB18030 的容量非常大，共有码位 160 万左右。另外，它与 GB13000 标准是兼容的。

因此，所有基于 GB13000 设计的软件都能够不经修改运行在支持 GB18030 的系统平台上。

UCS 与 UTF

国际标准 ISO 10646 定义了通用字符集 (Universal Character Set, UCS)。UCS 是所有其他字符集标准的一个超集. 它保证与其他字符集是双向兼容的. 就是说, 如果你将任何文本字符串翻译到 UCS 格式, 然后再翻译回原编码, 你不会丢失任何信息。

其编码方式类似于前文写到的 GB13000，其实是 GB13000 类似它才对。

因为迄今为止只分配了前 65534 个码位 (0x0000 到 0xFFFD)，故常常只要只用到基本多文种平面 BMP，只要两字节即可，即为 UCS-2。UCS-4 仅仅是在 UCS-2 的前面加上两字节的 0 而已。

历史上存在两个试图独立设计 Unicode 的组织，即国际标准化组织（ISO）和一个软件制造商的协会（unicode.org）。ISO 开发了 ISO 10646 项目，Unicode 协会开发了 Unicode 项目。

在 1991 年前后，双方都认识到世界不需要两个不兼容的字符集。于是它们开始合并双方的工作成果，并为创立一个单一编码表而协同工作。从 Unicode2.0 开始，Unicode 项目采用了与 ISO 10646-1 相同的字库和字码。

UCS 规定了怎么用多个字节表示各种文字。怎样传输这些编码，是由 UTF(UCS Transformation Format)规范规定的，常见的 UTF 规范包括 UTF-8、UTF-16、UTF-32。

在 Unix 下使用 UCS-2 (或 UCS-4) 会导致非常严重的问题. 用这些编码的字符串会包含一些特殊的字符, 比如 '' 或 '/', 它们在 文件名和其他 C 库函数参数里都有特别的含义. 另外, 大多数使用 ASCII 文件的 UNIX 下的工具, 如果不进行重大修改是无法读取 16 位的字符的. 基于这些原因, 在文件名, 文本文件, 环境变量等地方, UCS-2 不适合作为 Unicode 的外部编码.

在 ISO 10646-1 Annex R 和 RFC 2279 里定义的 UTF-8 编码没有这些问题. 它是在 Unix 风格的操作系统下使用 Unicode 的明显的方法。

UTF-8 以 8 位为单元对 UCS 进行编码。具有一下特性:

* UCS 字符 U+0000 到 U+007F (ASCII) 被编码为字节 0x00 到 0x7F (ASCII 兼容)。这意味着只包含 7 位 ASCII 字符的文件在 ASCII 和 UTF-8 两种编码方式下是一样的。

* 所有 >U+007F 的 UCS 字符被编码为一个多个字节的串, 每个字节都有标记位集。因此, ASCII 字节 (0x00-0x7F) 不可能作为任何其他字符的一部分。

* 表示非 ASCII 字符的多字节串的第一个字节总是在 0xC0 到 0xFD 的范围里, 并指出这个字符包含多少个字节. 多字节串的其余字节都在 0x80 到 0xBF 范围里. 这使得重新同步非常容易, 并使编码无国界, 且很少受丢失字节的影响。

* 可以编入所有可能的 2^{31} 个 UCS 代码。

* UTF-8 编码字符理论上可以最多到 6 个字节长, 然而 16 位 BMP 字符最多只用到 3 字节长。

* Bigendian UCS-4 字节串的排列顺序是预定的。

* 字节 0xFE 和 0xFF 在 UTF-8 编码中从未用到。

下列字节串用来表示一个字符。用到哪个串取决于该字符在 Unicode 中的序号。

U-00000000 – U-0000007F:	0*xxxxxxx*
U-00000080 – U-000007FF:	110*xxxxx* 10*xxxxxx*
U-00000800 – U-0000FFFF:	1110*xxxx* 10*xxxxxx* 10*xxxxxx*
U-00010000 – U-001FFFFF:	11110*xxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
U-00200000 – U-03FFFFFF:	111110*xx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
U-04000000 – U-7FFFFFFF:	1111110*x* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*

xxx 的位置由字符编码数的二进制表示的位填入。越靠右的 x 具有越少的特殊意义。只用最短的那个足够表达一个字符编码数的多字节串。注意在多字节串中，第一个字节的开头"1"的数目就是整个串中字节的数目。

例如: UCS 字符 U+00A9 = 1010 1001 (版权符号) 在 UTF-8 里的编码为:

11000010 10101001 = 0xC2 0xA9

而字符 U+2260 = 0010 0010 0110 0000 (不等于) 编码为:

11100010 10001001 10100000 = 0xE2 0x89 0xA0

字节序问题

这里不得不再提一下字节序问题，UTF-8 以字节为编码单元，没有字节序的问题。UTF-16 以两个字节为编码单元，在解释一个 UTF-16 文本前，首先要弄清楚每个编码单元的字节序。例如收到一个“奎”的 Unicode 编码是 594E，“乙”的 Unicode 编码是 4E59。如果我们收到 UTF-16 字节流“594E”，那么这是“奎”还是“乙”？

Unicode 规范中推荐的标记字节顺序的方法是 BOM，Byte Order Mark：

在 UCS 编码中有一个叫做"ZERO WIDTH NO-BREAK SPACE"的字符，它的编码是 FEFF。而 FFFE 在 UCS 中是不存在的字符，所以不应该出现在实际传输中。UCS 规范建议我们在传输字节流前，先传输字符"ZERO WIDTH NO-BREAK SPACE"。

这样如果接收者收到 FEFF，就表明这个字节流是 Big-Endian 的；如果收到 FFFE，就表明这个字节流是 Little-Endian 的。因此字符"ZERO WIDTH NO-BREAK SPACE"又被称作 BOM。

UTF-8 不需要 BOM 来表明字节顺序，但可以用 BOM 来表明编码方式。字符"ZERO WIDTH NO-BREAK SPACE"的 UTF-8 编码是 EF BB BF。所以如果接收者收到以 EF BB BF 开头的字节流，就知道这是 UTF-8 编码了。

Windows 就是使用 BOM 来标记文本文件的编码方式的。

三、输出问题

这个问题相对上面两个问题要简单的多，就是把内码映射到方块字上。从数学的角度看，这是一个双射函数。一个字符集里所有字符的形状描述信息集合在一起称为字形信息库，简称字库。不同的字体对应着不同的字库。在输出每一个汉字的时候，计算机都要先到字库中去找它的字形描述信息，然后把字形信息发送到相应的设备输出。

字库实在是太多了，一个个人的计算机中，常常有数十甚至上百种字体，字体收集狂人更是成套成套的收集……归根究底是因为只有这个函数影响人类的审美观。比起和计算机的亲近程度来，显然更贴近人一点。

汉字字形的描述方法之一是字模点阵码，用方阵表示汉字，描述简单，1 表黑点，0 表空白，堆成方阵即可。但一改变大小就失真。就跟放大缩小一般格式的图片一样。

另一种是轮廓描述法，用数学公式或者向量的方法刻画汉字的一笔一划。目前有的标准是 AdobeType1 和 True Type。这种方式精度相对要高得多，可以任意改变大小。可以类比一下矢量格式的图片。

ClearType 最近流行起来，它是一种通过使屏幕字体的边缘平滑来增强显示使字体清晰的技术。。 ClearType 尤其适用于液晶显示 (LCD) 设备，包括平面屏幕监视器和便携式计算机。

现有的很多汉字字体不支持 ClearType，想要“享受”的话，可以设置默认字体为微软雅黑。顺便说句，我受不了那玩意……

总结：

写这篇论文的过程实际上是我学习的过程，看了很多资料，知道了很多，从数学的角度简单地看待了一下三个问题，但是由于并不是要我自己去创造这些对应关系，仅仅只是学习研究，并没有从应具有的性质角度对其进行探讨。文中第二部分引用了很多文章，主要原因还是因为这是一个标准问题，不是我一个人随随便便就能制定的，只能做个“介绍”，无法去“探讨”了。

参考资料一览

《汉字编码设计学》潘德孚、詹振权

《计算机组成与结构》（第 2 版）徐福培

《汉字编码字符集的现状和发展》万加雷、陶晓鹏

《Linux 程序员必读：中文化与 GB18030 标准》作者未知

《GB 18030 介绍及其与相关标准的比较》NITS 技术文件 http://www.nits.gov.cn

《谈谈 Unicode 编码，简要解释 UCS、UTF、BMP、BOM 等名词》fmddlmyy

《UTF-8 and Unicode FAQ》Markus Kuhn [译]中国 LINUX 论坛翻译小组 xLoneStar