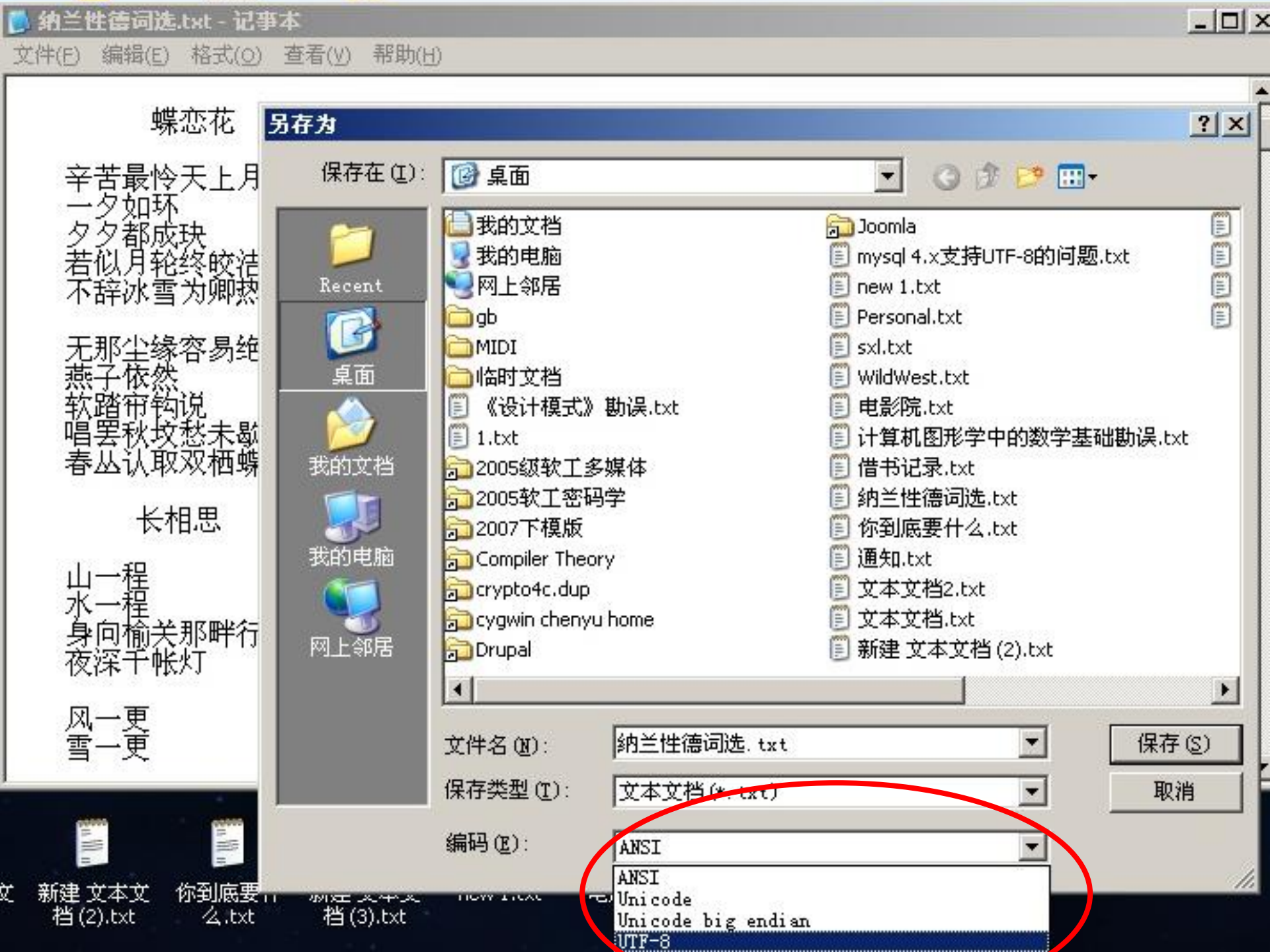
《Web程序设计》阅读材料——文字的编码介绍

福州大学　数计学院
软件工程系　　陈　昱

文字的编码
Character Encodings

文字在计算机内的存储形式


纳兰性德词选.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
蝶恋花
辛苦最怜天上月
一夕如环
夕夕都成玦
若似月轮终皎洁
不辞冰雪为卿热
无那尘缘容易绝
燕子依然
软踏帘钩说
唱罢秋坟愁未歇
春丛认取双栖蝶
长相思
山一程
水一程
身向榆关那畔行
夜深千帐灯
风一更
雪一更
另存为
保存在(I):
桌面
Recent
桌面
我的文档
我的电脑
网上邻居
我的文档
我的电脑
网上邻居
gb
MIDI
临时文档
《设计模式》勘误.txt
1.txt
2005级软工多媒体
2005软工密码学
2007下模版
Compiler Theory
crypto4c.dup
cygwin chenyu home
Drupal
Joomla
mysql 4.x支持UTF-8的问题.txt
new 1.txt
Personal.txt
sxl.txt
WildWest.txt
电影院.txt
计算机图形学中的数学基础勘误.txt
借书记录.txt
纳兰性德词选.txt
你到底要什么.txt
通知.txt
文本文档2.txt
文本文档.txt
新建 文本文档 (2).txt
文件名(N): 纳兰性德词选.txt
保存类型(T): 文本文档(*.txt)
编码(E): ANSI
保存(S)
取消
ANSI
Unicode
Unicode big endian
UTF-8
新建 文本文档 (2).txt
你到底要什么.txt
新建 文本文档 (3).txt

什么是字符集？

- 字符 (Character) 是文字与符号的总称，包括文字、图形符号、数学符号等。
- 一组抽象字符的集合就是字符集 (Charset)
- 字符集常常和一种具体的语言文字对应起来
  - 比如繁体汉字字符集、日文字符集

什么是编码？

- 计算机要处理各种字符，就需要将字符和二进制数对应起来，这种对应关系就是字符编码（Encoding）
- 制定编码首先要确定字符集，并将字符集内的字符排序，然后和二进制数字对应起来。
- 根据字符集内字符的多少，确定用几个字节来编码。

文字的编码

- 编码——文字在计算机中的表现形式
- 字符必须编码后才能被计算机处理。计算机使用的缺省编码方式就是计算机的内码。
- 常见编码
  - ASCII
  - GB 2312-80, GBK, GB 18030-2000
  - BIG-5
  - Unicode

ASCII

- 美国信息互换标准代码
- 现今最通用的单字节编码系统
  - 包含大小写英文字母、阿拉伯数字、各种标点符号及一些特殊的代码
- 实际为7位
  - 33个控制字符
  - 95个可打印字符

ASCII control characters

- 00 Null character 空字符
- 07 \a Bell 响铃
- 08 \b Backspace 退格
- 09 \t table 制表符
- 0A \n Line feed 换行
- 0D \r Carriage return 回车

换行与回车？

- PC (DOS, Windows)
  - 0D 0A (\r\n)
- Mac
  - 0D
- UNIX
  - 0A
  - printf("Hello World!\n");

汉字编码

- 为进行信息交换，各汉字使用地区都制订了一系列汉字字符集标准。
- 国标码
  - GB 2312, GBK, GB 18030
- Big5 码
- Unicode

GB 2312-80

- GB 2312-80
  - 简体中文字符集的中国国家标准
  - 全称为《信息交换用汉字编码字符集·基本集》
  - GB2312 编码通行于大陆；新加坡等地也采用此编码
  - 几乎所有的中文系统和国际化的软件都支持GB 2312

GB 2312-80 字符集大小

- GB 2312 标准共收录 6763 个汉字
  - 其中，一级汉字 3755 个，二级汉字 3008 个
  - 还包括其他语种的 682 个全角字符
  - 所收录的汉字已经覆盖 99.75% 的使用频率

GB 2312-80 字符集局限性

- 对于人名、古汉语等方面出现的罕用字，GB 2312不能处理：
  - –朱镕基　游錫堃　陶喆
- 这导致了后来GBK及GB 18030汉字字符集的出现

GB2312 编码方式：分区表示

- GB 2312 中对所收汉字进行了“分区”处理，每区含有 94 个汉字/符号。这种表示方式也称为“区位码”。
  - 01-09 区为特殊符号。
  - 16-55 区为一级汉字，按拼音排序。
  - 56-87 区为二级汉字，按部首/笔画排序。
  - 10-15 区及88-94区则未有编码。
- 例如：“啊”字是 GB2312 之中的第一个汉字，它的区位码就是 1601。

字节结构 (内码表示)

- 每个汉字及符号以两个字节来表示
  - 第一个字节称为“高位字节”，第二个字节称为“低位字节”。
  - “高位字节”使用了0xA1-0xF7(把01-87区的区号加上0xA0)
  - “低位字节”使用了0xA1-0xFE(把01-94加上0xA0)。
- 出于兼容 ASCII 的考虑

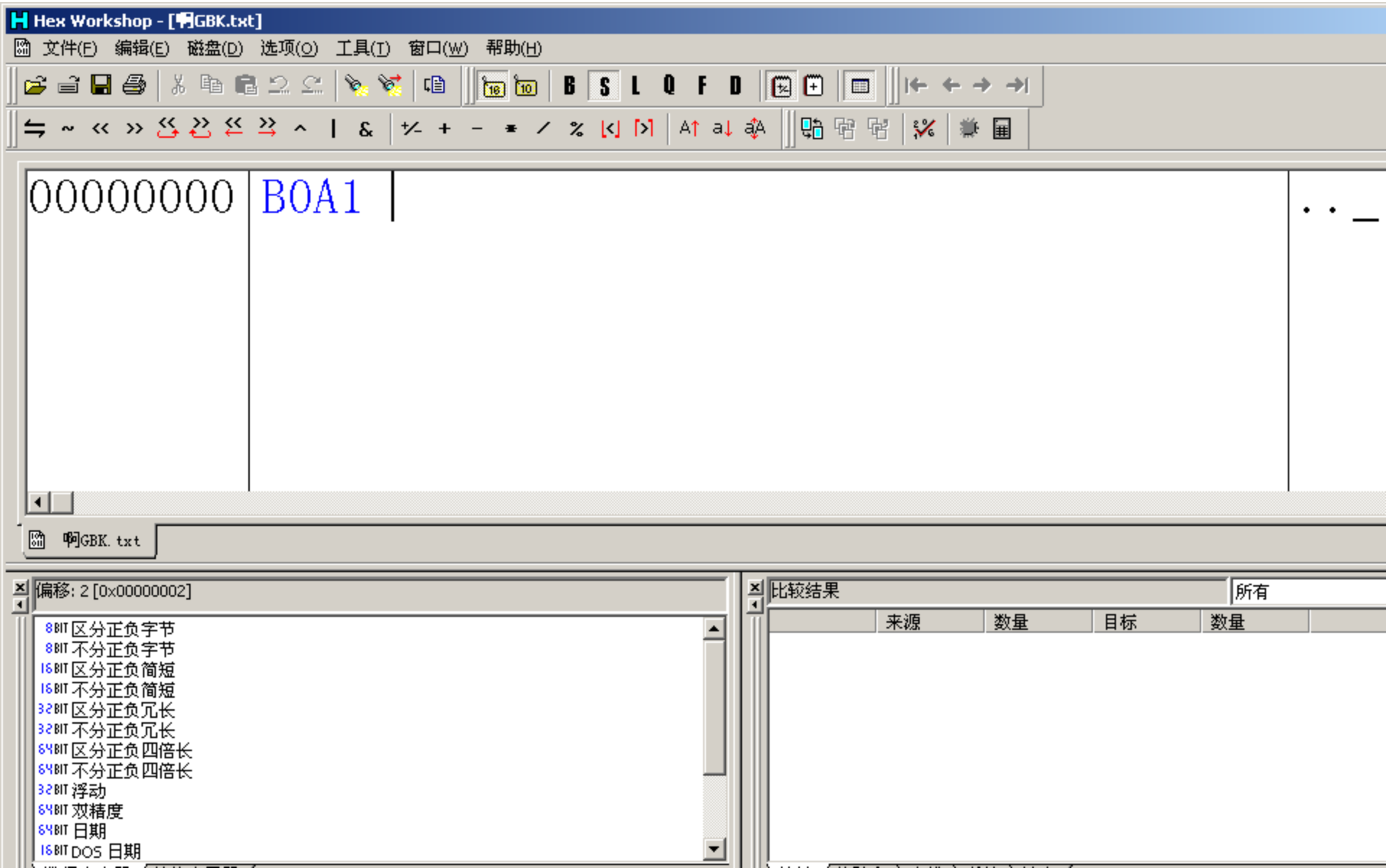
字节结构 (内码表示)

- 例如：
- "啊" 字在大多数程序中，会以0xB0A1储存（即所谓内码）

- 与区位码 [1601] 对比：

  0xB0=0xA0+16,　　0xA1=0xA0+1

HEXWorkshop



GBK 汉字内码扩展规范

- GBK 全名为汉字内码扩展规范。
- GBK 收录了所有 Unicode 1.1 及 GB 13000.1-93 之中的汉字，但是编码方式不同，仅仅是一个过渡方案。
- 总共有 20,902 个汉字

GBK 汉字内码扩展规范

- 最初是作为微软对 GB2312 的扩展出现在 Windows 95 简体中文版中
- 由于 Windows 产品的流行和在大陆广泛被使用，国家有关部门将其作为技术规范
- 注意：GBK并非国家正式标准，只是规范指导性文件

GB 18030-2000 主要特点

- GB 18030-2000 是最新的国家标准
  - 与 GB 2312-80 与 GBK 兼容
  - 采用多字节编码方式，每个字可以由1个、2个或4个字节组成
  - 编码空间庞大，最多可定义161万个字符
  - 支持中国国内少数民族的文字

GB 18030-2000

- 包含所有 GB 13000-93 及 Unicode 3.1 字符集中的字符
- 收录了藏文、蒙文、维吾尔文等中国主要的少数民族文字
- 一共收录了 27,484 个汉字

Windows 对 GB18030 的支持

- 微软提供了 GB18030 的升级包
- 但这个升级包只是提供了一套支持 CJK 扩展A的 6582 个汉字的新字体：新宋体-18030，并不改变内码。
- Windows 的内码仍然是 GBK
  - google: Microsoft GB18030 Support Package

编码之间的兼容性

- 从 ASCII、GB2312 到 GBK，再到 GB18030，这些编码方法是向下兼容的
- 所谓兼容，是指同一个字符在这些方案中总是有相同的编码，后面的标准支持更多的字符

编码之间的兼容性

- 比如 a，从 ASCII、GB2312、GBK 再到 GB18030，都是 0x61
- 比如"啊”　，都是 0xB0A1

编码之间的兼容性

- 按照程序员的称呼，GB2312、GBK 都属于双字节字符集 (DBCS)
- GB18030 的汉字编码采用双字节和4字节方案
- 4 字节编码的码位是新增加的 Unicode CJK 扩展A 的 6582 个汉字

编码之间的兼容性

- 在这些编码中，英文和中文可以统一地处理
- 区分中文编码的方法是高字节的最高位不为 0

程序处理

- GB2312 的两个字节的最高位都是1
- 但符合这个条件的编码只有 128*128=16384 个

- 所以 GBK 和 GB18030 的低字节最高位都可能不是 1

程序处理

- 不过这不影响双字符字符流的解析：
- 从左向右按字节读文件，碰到最高位为 0 的字节则为 ASCII，碰到最高位为 1，则连同后面的一个字节都属于双字符编码
- **0x FD93**

Big5 大五码

- Big5，又称为大五码或五大码
  - 繁体中文社群中最常用的电脑汉字字符集标准，共收录 13,053 个汉字
  - 虽普及于台湾、香港与澳门等繁体中文通行区，但并非当地的国家标准
  - 然而，由于 Windows、MacOS 等操作系统以及周边软件长期使用，Big5 已成为业界标准（de facto standard）

Big5 历史及名称

- Big5 是在1984年由中华民国财团法人资讯工业策进会和五间有意愿共同推动电脑中文化的资讯公司所共同创立，故称五大码。
  - 宏碁、神通、佳佳、零壹及大众
- 在 Big5 码诞生后，加上后来倚天中文系统的高度普及，使后来的微软 Windows 3.x 等亦予以采用。

Big5 使用范围

- 现在，除了台湾外，其他使用繁体汉字的地区，如香港、澳门，还有海外华人，都普遍使用 Big5 码
- 已经成为繁体中文显示的标准格式

Big5 字节结构

- Big5 码以两个字节来编码一个字
  - “高位字节”使用了 0x81-0xFE，
  - “低位字节”使用了 0x40-0x7E，及 0xA1-0xFE。

- Big5 重复地收录了两个相同的字：“兀、兀”、“殽、殽”，是一个 bug

Unicode 的原理

Unicode

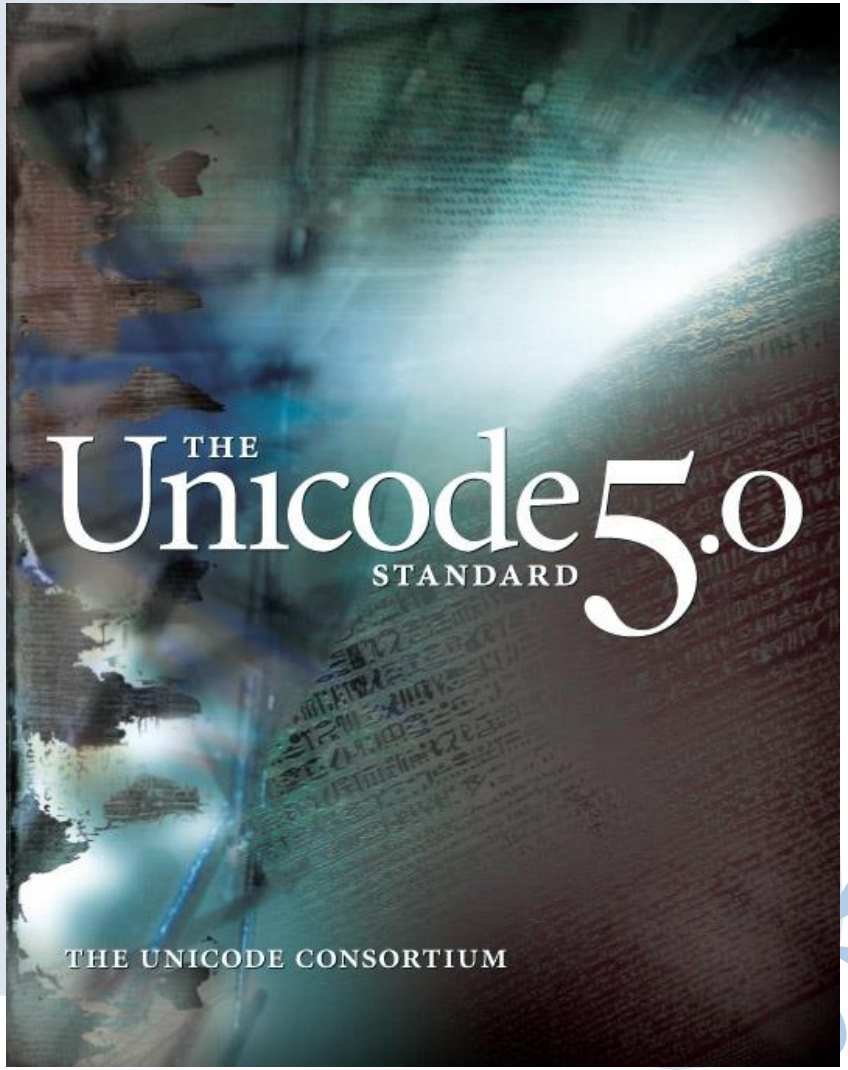
- Unicode：统一码、万国码、单一码
- 它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求

Unicode

- 1990年开始研发，1994年正式公布
- 随着计算机处理能力的增强，Unicode 也在面世以来的十多年里逐步得到普及
  - 比如像 Java, .NET 都以 Unicode 作为内部编码

Unicode 的版本

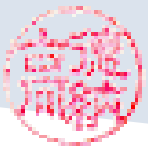
- 最新版本的 Unicode 是 2009 年10月1日推出的 Unicode 5.2
- http://www.unicode.org



Unicode 对 Web 的影响

is my Chinese name, given to me in 1977 by Frances Yao. Many browsers are now able to display these characters in their typeset form (高德纳); hopefully this will be routinely possible as **computers and Unicode help the world to get smaller.**

- Donald E. Knuth
  - http://www-cs-faculty.stanford.edu/~knuth/faq.html#asian



Unicode 简介

- Unicode：统一码
- 顾名思义是将世界各国几十种混乱的编码整合在一起的努力
- 通常每个字符在 Unicode 中用2个字节表示。这样理论上一共最多可以表示 2^{16} 即 65536 个字符，即所谓 USC-2

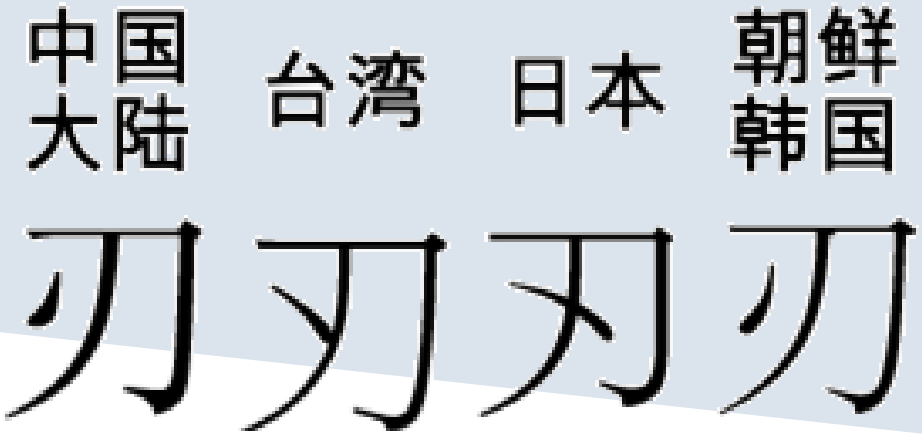
Unihan 统汉字

- USC-2 对于汉字来说捉襟见肘（光康熙字典就有四万七）

- 为此，Unicode 采用所谓“中日韩文整合”的解决方案，将中日韩文中笔划近似的汉字，尽量以一个编码来代表。

Unihan 统汉字

- 比如“草字头”在繁体中是四划，在简体和日文中是三划。
- Unicode 忽略差别，不再替所有带四划草字头的字单独编码。



Unihan 统汉字

- 如果字型差异比较大，比如言字旁，就分别编码，一简一繁。
- 经过“中日韩文整合”的汉字，在 Unicode 中称作 Unihan。共有两万多个统汉字。
- ftp://ftp.unicode.org/Public/UNIDATA/Unihan.zip

Unicode 的编码和实现

- Unicode 编码系统可分为编码方式和实现方式两个层次
- 编码方式：在字符集中用几个字节来编码一个字符
  - 字节数决定了最多能包含多少字符
  - 一个字符在 Unicode 字符集中的编码是确定的

Unicode 的编码和实现

- 实现方式：一个 Unicode 字符在系统中的实际存储方式
- 由于不同系统平台的设计不一定一致，以及出于节省空间的目的，对 Unicode 编码的实现方式有所不同

Unicode 编码方式

- Unicode Character Set，简称为UCS
  - UCS 只是规定如何编码，并没有规定如何传输、保存这个编码。
- UCS-2：目前实际应用的编码方式
  - 每个字符 2 个字节， 2^{16} 个字符
- UTF-32/UCS-4：未来的编码方式
  - 每个字符 4 字节，最多能表示 2^{31} 个字符

Unicode 编码方式

- UCS-2 里的字符构成"基本多文种平面"
  - Basic Multilingual Plane, 简称 BMP
- BMP 字符的 Unicode 编码表示为 U+***hhhh***
- “汉”字的UCS编码是 0x 6C49

Unicode 实现方式

- UCS 只是规定如何编码，并没有规定如何传输、保存这个编码
- 例如"汉"字的UCS编码是 0x6C49
  - 可以用 4个 ASCII 字符来传输、保存这个编码：字符串 “6C49”
  - 也可以用 UTF-8 编码: 3个连续的字节 E6 B1 89 来表示它

Unicode 实现方式

- 关键在于通信双方都要认可
- Unicode 的实现方式称为 *Unicode转换格式*
  - ***Unicode Translation Format，UTF***
- 比较常用的有
  - UTF-8
  - UTF-16

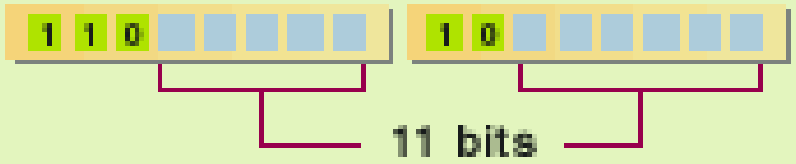
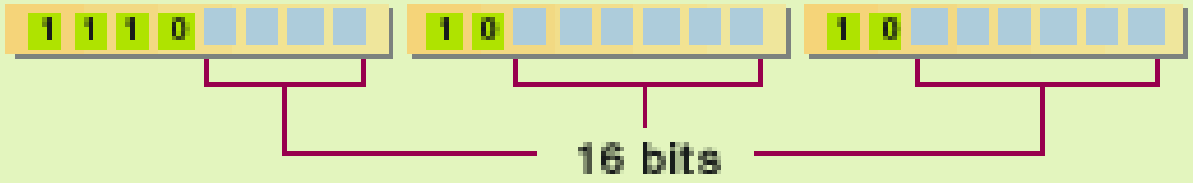
UTF-8

- 一种变长编码
- 它将7位基本ASCII字符仍用7位编码表示，占用一个字节（首位补0）
- 遇到其他 Unicode 字符的情况，将按一定算法转换，每个字符使用1～3个字节编码，并利用首位为0或1进行识别。

UTF-8 编码原理

- UTF-8 就是以8位为单元对UCS进行编码
- 从UCS-2到UTF-8的编码方式如下：
- UCS-2编码 (16进制) UTF-8 (二进制)
  - 0000 - 007F 0xxxxxxx
  - 0080 - 07FF 110xxxxx 10xxxxxx
  - 0800 - FFFF 1110xxxx 10xxxxxx 10xxxxxx

UTF-8 编码原理

Unicode 中的區位	排列格式
U+0000 ~ U+007E	
U+0080 ~ U+07FF	
U+0800 ~ U+FFFF	

UTF-8 编码举例

- 例如”汉“字的 Unicode 编码 (UCS-2) 是 6C49
- 6C49 在 0800-FFFF 之间，所以肯定要用 3 字节模板：

  1110xxxx 10xxxxxx 10xxxxxx

UTF-8 编码举例

- 将 6C49 写成二进制是：

  01101100 01001001

- 用这个比特流依次代替模板中的 x，得到：

  11100110　10110001　10001001

- 即 E6 B1 89

UTF-8 的优点

- 采用字节编码，与ASCII兼容
- 字与字之间容易划分：
  - 0, 110, 1110, 11110 ……

UTF-8 的缺点

- 除了英语，其他语种的文件变“胖”了。
- 在旧的中文、日文及韩文编码之中，每字符都使用 2 字节储存，而UTF-8 须使用 3 字节。最坏情况变成原来 150%
- 泰语以往使用的 ISO 8859-11，每字符只使用 1 字节储存，而 UTF-8 须使用 3 字节（更惨～～～）

UTF-16

- UTF-16 编码直接使用 Unicode 编码（仅限于 BMP 字符）
- 无论是拉丁字母、汉字或其他文字或符号），一律使用 2 *字节* 储存

问题来了

- 由于每个字符占用了两个字节，在 Macintosh 机和 PC 机上对字节顺序的理解是不一致的
- 这时同一字节流可能会被解释为不同内容

问题来了

- 如：
- 编码为 U+594E 的字符“奎”与编码为 U+4E59 的“乙”
- 就可能发生混淆

解决方案

- 于是在 UTF-16 编码实现方式中使用了
  - 大尾序（big-endian）、小尾序（little-endian）的概念
  - 以及 BOM（Byte Order Mark）解决方案

Big Endian 和 Little Endian

- Big Endian 和 Little Endian 是CPU处理多字节数据的不同方式。

- 例如“汉”字的 Unicode 编码是 6C 49。那么写到文件里时，究竟是将 6C写在前面，还是将 49 写在前面？

Big Endian 和 Little Endian

- 如果将 6C 写在前面，就是 Big Endian。
- 如果将 49 写在前面，就是 Little Endian。
- 一般将 Endian 翻译成 “字节序” ，，将 big endian 和 little endian 称作“大尾序”和“小尾序” 。

Big Endian 和 Little Endian

- Big Endian machine: 它认为它读到的第一个字节是最高位字节
- Little Endian machine: 它认为它读到的第一个字节是最低位字节
- 例如，从内存地址0x0000开始有以下数据
  0x0000 0x12
  0x0001 0x34
  0x0002 0xab
  0x0003 0xcd

Big Endian 和 Little Endian

- Little-Endian 小尾序
  - Intel x86
- Big-Endian 大尾序
  - IBM Power-PC
  - SUN SPARC
  - SGI MIPS
  - Moto 68k

Byte Order Mark (BOM)

- 为了标识字节序，在存储和传输字节流之前，先通过一个标识告诉对方计算机所用的字节序
- 这就是 BOM

Byte Order Mark (BOM)

- 为了标识字节序， UCS 规范建议我们在传输字节流前，先传输字符 0xFEFF
  - 在 UCS 编码中有一个叫做"ZERO WIDTH NO-BREAK SPACE" 的特殊字符，它的编码是 0xFEFF
  - 而 0xFFFE 在 UCS 中是不存在的字符，所以不应该出现在实际传输中

BOM

- 这样如果接收者
  - 收到 0xFEFF，就表明这个字节流是 Big-Endian
  - 收到 0xFFFE，就表明这个字节流是 Little-Endian
- 因此字符 “ZERO WIDTH NO-BREAK SPACE” 又被称作 BOM

BOM

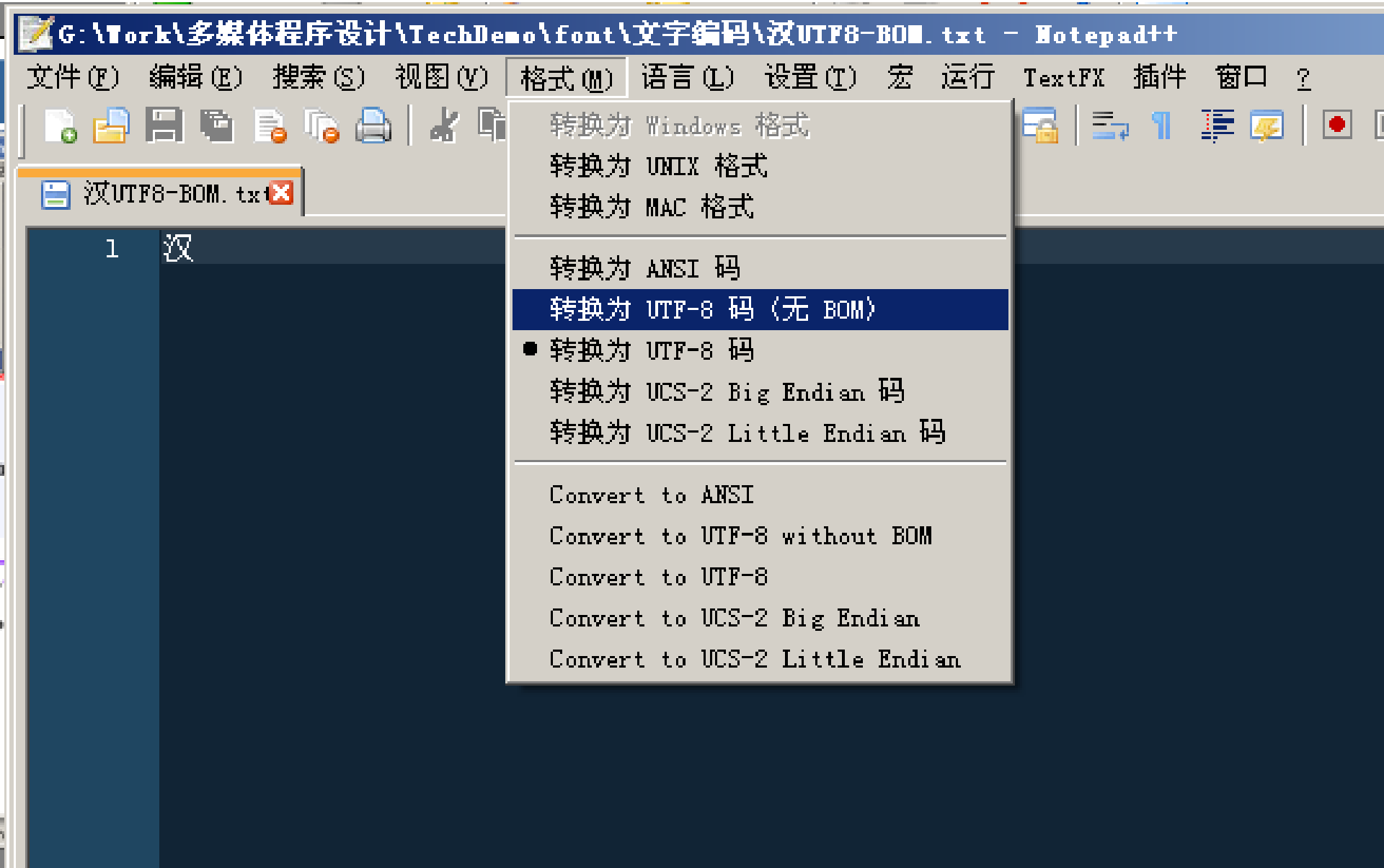
- 以下的例子有三个字符：“朱”、半角逗号、“聿”

使用 UTF-16 编码的例子						
编码名称	编码次序	编码				
UTF-16LE	小尾序		31 67	2C 00	7F 80	
UTF-16BE	大尾序		67 31	00 2C	80 7F	
UTF-16	小尾序，包含BOM	FF FE	31 67	2C 00	7F 80	
UTF-16	大尾序，包含BOM	FE FF	67 31	00 2C	80 7F	

UTF-8 与 BOM

- UTF-8 不需要BOM来表明字节顺序(why?)，但可以用BOM来表明编码方式。
- 字符 0xFEFF 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 可以使用 BOM 来识别文本文件的编码方式

UTF-8 与 BOM （notepad++）



Windows 的内码，代码页

内码

- 内码是指操作系统内部的字符编码
- 现在的 Windows 在系统内部支持 Unicode，然后用代码页适应各种语言，“内码”的概念就比较模糊了

代码页

- 所谓代码页 (code page) 就是针对一种语言文字的字符编码
- Windows 的内码是 Unicode，它在技术上可以同时支持多个代码页
- 微软一般将缺省代码页指定的编码说成是内码

代码页

- 例如
  - GBK 的 code page 是 CP936
  - BIG5 的 code page 是 CP950
  - GB2312 的 code page 是CP20936
  - http://msdn.microsoft.com/en-us/library/aa288104.aspx

代码页

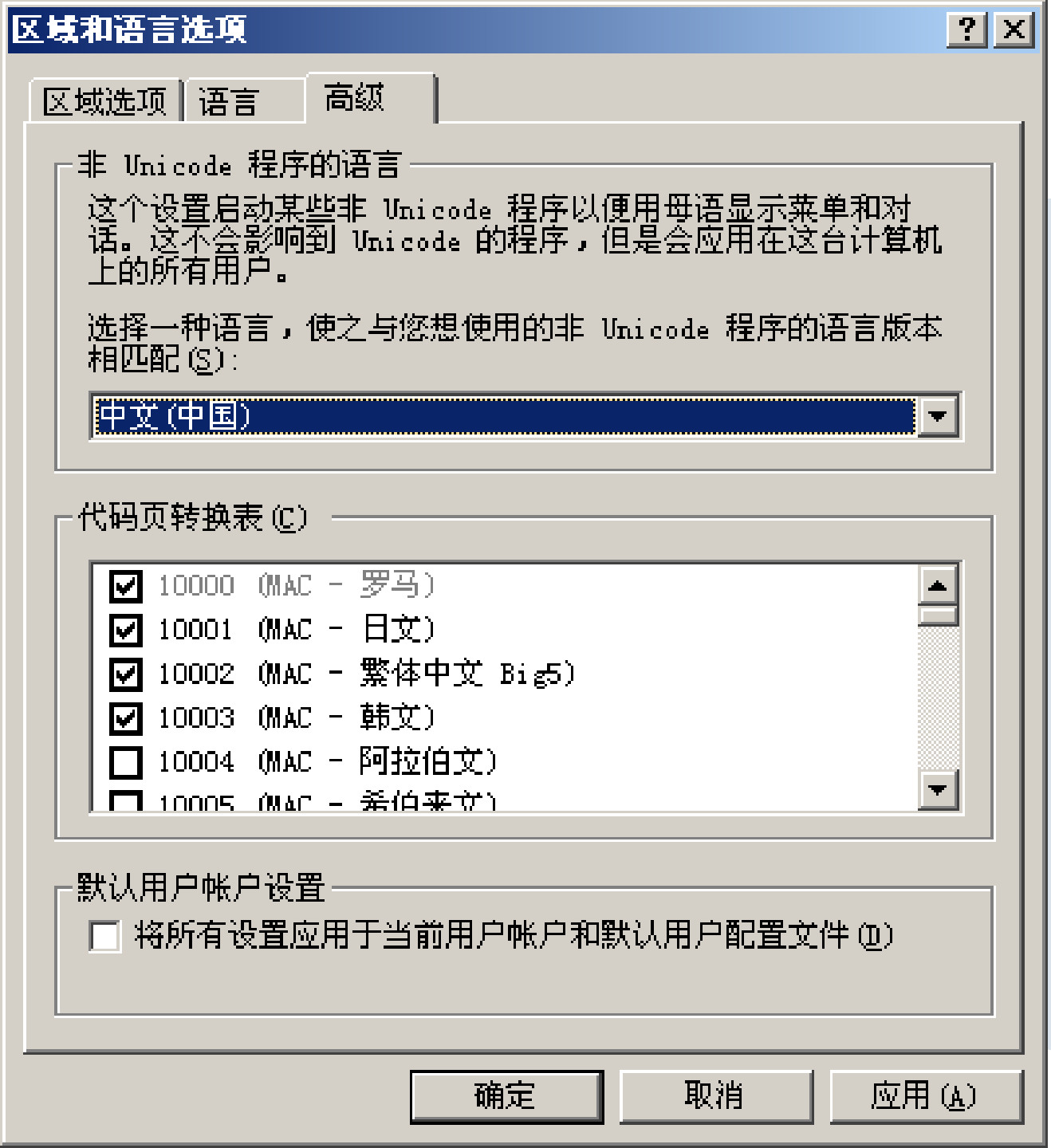
- Windows 中有缺省代码页的概念，即缺省用什么编码来解释字符
- 例如 Windows 的记事本打开了一个文本文件，里面的内容是字节流：BA、BA、D7、D6
- Windows 应该去怎么解释它呢？

乱码

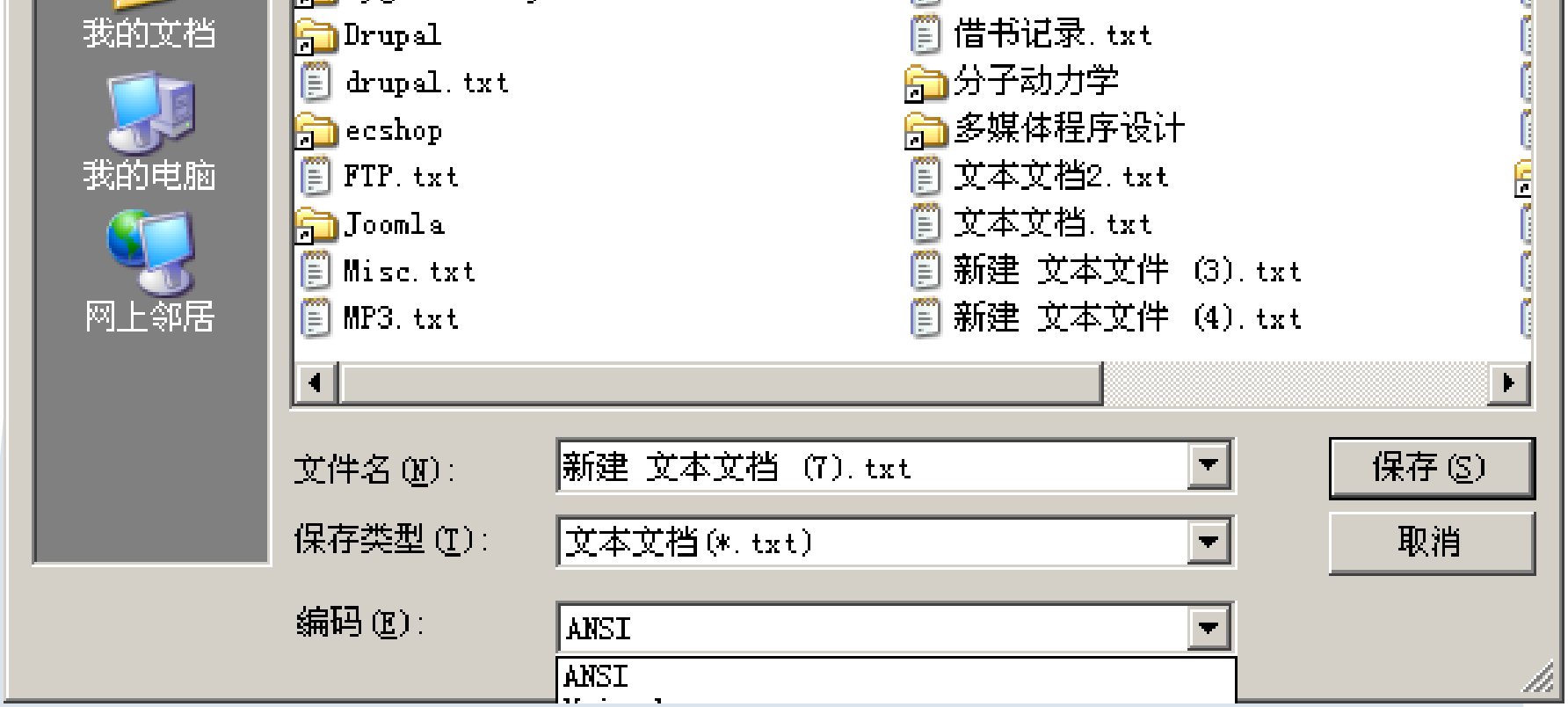
- 如果按GBK去解释，就会得到 “汉字” 两个字。
- 按照其它编码解释，可能找不到对应的字符，也可能找到错误的字符。
- 也就是乱码了：messy code
- Unicode 规定，找不到正确对应的字，一律换成问号 “？”

Windows 识别编码的方式（1）

- Windows 按照当前的缺省代码页设置去解释文本文件里的字节流
- 缺省代码页可以通过控制面板的“区域与语言选项”对话框中的“高级”设置


区域和语言选项
区域选项
语言
高级
非 Unicode 程序的语言
这个设置启动某些非 Unicode 程序以便用母语显示菜单和对话。这不会影响到 Unicode 的程序，但是会应用在这台计算机上的所有用户。
选择一种语言，使之与您想使用的非 Unicode 程序的语言版本相匹配(S):
中文(中国)
代码页转换表(C)
10000 (MAC - 罗马)
10001 (MAC - 日文)
10002 (MAC - 繁体中文 Big5)
10003 (MAC - 韩文)
10004 (MAC - 阿拉伯文)
默认用户帐户设置
将所有设置应用于当前用户帐户和默认用户配置文件(D)
确定
取消
应用(A)

区域与语言选项



- 记事本的“另存为”中有一项 ANSI，其实就是按照缺省代码页的编码方法保存

Windows 识别编码的方式（2）

- 如果文件能说明自己使用什么编码，用户又安装了对应的代码页字体，Windows 就能正确显示：
  - 例如： TXT 文件头部的 BOM
  - 例如： <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

小结

- 文字：最重要的信息传播媒体
  - 汉字的基础知识
- 文字的编码: 文字在计算机中的存储形式
  - ASCII
  - GB 2312, GBK, GB 18030
  - Big5
  - Unicode