[UTF-8和中文字符编码（GB2312、GBK、GIB5、GB18030）的识别](http://blog.chinaunix.net/xmlrpc.php?r=blog/article&uid=29594401&id=4257356) 2014-05-16 20:09:50

分类： LINUX

1、编码方式介绍

了解一种[字符集](http://baike.baidu.com/view/51987.htm)编码主要是要了解该编码的编码范围，编码对应的字符集（都包含哪些字符），和其他字符集编码之间的关系等。

ASCII

ASCII码是7位编码，编码范围是0x00-0x7F。ASCII[字符集](http://baike.baidu.com/view/51987.htm)包括英文字母、阿拉伯数字和标点符号等[字符](http://baike.baidu.com/view/263416.htm)。其中0x00-0x1F和0x7F共33个[控制字符](http://baike.baidu.com/view/1112575.htm)。[1]

只支持ASCII码的系统会忽略每个[字节](http://baike.baidu.com/view/60408.htm)的最高位，只认为低7位是有效位。HZ[字符编码](http://baike.baidu.com/view/1204863.htm)就是早期为了在只支持7位ASCII系统中传输中文而设计的编码。早期很多[邮件系统](http://baike.baidu.com/view/1677840.htm)也只支持[ASCII编码](http://baike.baidu.com/view/3001329.htm)，为了传输中文邮件必须使用BASE64或者其他编码方式。  
utf-8  
utf-8编码的判断格式如下：  
1字节 0xxxxxxx  
2字节 110xxxxx 10xxxxxx  
3字节 1110xxxx 10xxxxxx 10xxxxxx  
4字节 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx  
5字节 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx  
6字节 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx  
这是标准的utf-8编码格式

GB2312

GB2312是基于区位码设计的，区位码把[编码表](http://baike.baidu.com/view/421804.htm)分为94个区，每个区对应94个位，每个字符的区号和位号组合起来就是该汉字的区位码。区位码一般 用10进制数来表示，如1601就表示16区1位，对应的字符是“啊”。在区位码的区号和位号上分别加上0xA0就得到了GB2312编码。

**区位码中01-09区是符号、数字区，16-87区是汉字区**，10-15和88-94是未定义的空白区。它将收录的汉字分成两级：第一级是常用汉字 计3755个，置于16-55区，按[汉语拼音字母](http://baike.baidu.com/view/104417.htm)/笔形顺序排列；第二级汉字是次常用汉字计3008个，置于56-87区，按部首/笔画顺序排列。一级汉 字是按照拼音排序的，这个就可以得到某个拼音在一级汉字区位中的范围，很多根据汉字可以得到拼音的程序就是根据这个原理编写的。

GB2312[字符集](http://baike.baidu.com/view/51987.htm)中除常用[简体汉字](http://baike.baidu.com/view/2691330.htm)字符外还包括希腊字母、日文平假名及片假名字母、俄语[西里尔字母](http://baike.baidu.com/view/701246.htm)等字符，未收录[繁体中文](http://baike.baidu.com/view/148357.htm)汉字和一些生僻字。可以用繁体汉字测试某些系统是不是只支持GB2312编码。

GB2312的编码范围是0xA1A1-0xFEFE，去掉未定义的区域之后可以理解为实际编码范围是0xA1A1-0xF7FE（注意：实际编码值是前面的区、位值 + A0）。

GBK

GBK编码是GB2312编码的超集，向下完全兼容GB2312，同时GBK收录了Unicode[基本多文种平面](http://baike.baidu.com/view/628163.htm)中的所有CJK汉字。同 GB2312一样，GBK也支持希腊字母、日文假名字母、俄语字母等[字符](http://baike.baidu.com/view/263416.htm)，但不支持韩语中的表音字符（非汉字字符）。GBK还收录了GB2312不包含的 汉字部首符号、竖排标点符号等字符。

GBK的整体编码范围是为0x8140-0xFEFE，不包括低[字节](http://baike.baidu.com/view/60408.htm)是0×7F的组合。高[字节](http://baike.baidu.com/view/60408.htm)范围是0×81-0xFE，低字节范围是0x40-7E和0x80-0xFE。

低[字节](http://baike.baidu.com/view/60408.htm)是0x40-0x7E的GBK[字符](http://baike.baidu.com/view/263416.htm)有一定特殊性，因为这些字符占用了ASCII码的位置，这样会给一些系统带来麻烦。

有些系统中用0x40-0x7E中的[字符](http://baike.baidu.com/view/263416.htm)（如“|”）做特殊符号，在定位这些符号时又没有判断这些符号是不是属于某个 GBK字符的低[字节](http://baike.baidu.com/view/60408.htm)，这样就会造成错误判断。在支持GB2312的环境下就不存在这个问题。需要注意的是支持GBK的环境中小于0x80的某个[字节](http://baike.baidu.com/view/60408.htm)未必就 是ASCII符号；另外就是最好选用小于0×40的ASCII符号做一些特殊符号，这样就可以快速定位，且不用担心是某个汉字的另一半。Big5编码中也存在相应问题。

CP936和GBK的有些许差别，绝大多数情况下可以把CP936当作GBK的别名。

GB18030

GB18030编码向下兼容GBK和GB2312，兼容的含义是不仅[字符](http://baike.baidu.com/view/263416.htm)兼容，而且相同字符的编码也相同。GB18030收录了所有Unicode3.1中的字符，包括[中国少数民族](http://baike.baidu.com/view/1945582.htm)字符，GBK不支持的韩文字符等等，也可以说是世界大多民族的文字符号都被收录在内。

GBK和GB2312都是双[字节](http://baike.baidu.com/view/60408.htm)等宽编码，如果算上和ASCII兼容所支持的单[字节](http://baike.baidu.com/view/60408.htm)，也可以理解为是单字节和双字节混合的[变长编码](http://baike.baidu.com/view/1486307.htm)。GB18030编码是[变长编码](http://baike.baidu.com/view/1486307.htm)，有单字节、双字节和四字节三种方式。

GB18030的单[字节](http://baike.baidu.com/view/60408.htm)编码范围是0x00-0x7F，完全等同与ASCII；双字节编码的范围和GBK相同，高字节是0x81-0xFE，低字节 的编码范围是0x40-0x7E和0x80-FE；四字节编码中第一、三字节的编码范围是0x81-0xFE，二、四字节是0x30-0x39。

BIG5

Big5是双[字节](http://baike.baidu.com/view/60408.htm)编码，高字节编码范围是0x81-0xFE，低字节编码范围是0x40-0x7E和0xA1-0xFE。和GBK相比，少了低[字节](http://baike.baidu.com/view/60408.htm)是0x80-0xA0的组合。0x8140-0xA0FE是保留区域，用于用户造字区。

Big5收录的汉字只包括繁体汉字，不包括[简体汉字](http://baike.baidu.com/view/2691330.htm)，一些生僻的汉字也没有收录。GBK收录的日文假名[字符](http://baike.baidu.com/view/263416.htm)、俄文字符Big5也没有收录。因为Big5当中收录的[字符](http://baike.baidu.com/view/263416.htm)有限，因此有很多在Big5基础上扩展的编码，如[倚天中文系统](http://baike.baidu.com/view/1186762.htm)。Windows系统上使用的[代码页](http://baike.baidu.com/view/684997.htm)CP950也可以理解为是对Big5的扩展，在Big5的基础上增加了7个汉字和一些符号。Big5编码对应的[字符集](http://baike.baidu.com/view/51987.htm)是GBK字符集的子集，也就是说Big5收录的字符是GBK收录字符的一部分，但相同字符的编码不同。

因为Big5也占用了ASCII的编码空间（低[字节](http://baike.baidu.com/view/60408.htm)所使用的0x40-0x7E），所以Big5编码在一些环境下存在和GBK编码相同的问题，即低字节范围为0x40-0x7E的[字符](http://baike.baidu.com/view/263416.htm)有可能会被误处理，尤其是低字节是0x5C（"/"）和0x7C（"|"）的字符。可以参考GBK一节相应说明。

尽管有些区别，大多数情况下可以把CP950当作Big5的别名。  
  
2、识别编码

**点击(此处)折叠或打开**

1. //judge the byte whether begin with binary 10
2. int Encoder::is\_utf8\_special\_byte(unsigned char c)
3. {
4. unsigned special\_byte = 0X02; //binary 00000010
5. if (c >> 6 == special\_byte) {
6. return 1;
7. } else {
8. return 0;
9. }
10. }
11. int Encoder::is\_utf8\_code(const string& str)
12. {
13. unsigned one\_byte = 0X00; //binary 00000000
14. unsigned two\_byte = 0X06; //binary 00000110
15. unsigned three\_byte = 0X0E; //binary 00001110
16. unsigned four\_byte = 0X1E; //binary 00011110
17. unsigned five\_byte = 0X3E; //binary 00111110
18. unsigned six\_byte = 0X7E; //binary 01111110
20. int utf8\_yes = 0;
21. int utf8\_no = 0;
23. unsigned char k = 0;
24. unsigned char m = 0;
25. unsigned char n = 0;
26. unsigned char p = 0;
27. unsigned char q = 0;
29. unsigned char c = 0;
30. for (uint i=0; i<str.size();) {
31. c = (unsigned char)str[i];
32. if (c>>7 == one\_byte) {
33. i++;
34. continue;
35. } else if (c>>5 == two\_byte) {
36. k = (unsigned char)str[i+1];
37. if ( is\_utf8\_special\_byte(k) ) {
38. utf8\_yes++;
39. i += 2;
40. continue;
41. }
42. } else if (c>>4 == three\_byte) {
43. m = (unsigned char)str[i+1];
44. n = (unsigned char)str[i+2];
45. if ( is\_utf8\_special\_byte(m)
46. && is\_utf8\_special\_byte(n) ) {
47. utf8\_yes++;
48. i += 3;
49. continue;
50. }
51. } else if (c>>3 == four\_byte) {
52. k = (unsigned char)str[i+1];
53. m = (unsigned char)str[i+2];
54. n = (unsigned char)str[i+3];
55. if ( is\_utf8\_special\_byte(k)
56. && is\_utf8\_special\_byte(m)
57. && is\_utf8\_special\_byte(n) ) {
58. utf8\_yes++;
59. i += 4;
60. continue;
61. }
62. } else if (c>>2 == five\_byte) {
63. unsigned char k = (unsigned char)str[i+1];
64. unsigned char m = (unsigned char)str[i+2];
65. unsigned char n = (unsigned char)str[i+3];
66. unsigned char p = (unsigned char)str[i+4];
67. if ( is\_utf8\_special\_byte(k)
68. && is\_utf8\_special\_byte(m)
69. && is\_utf8\_special\_byte(n)
70. && is\_utf8\_special\_byte(p) ) {
71. utf8\_yes++;
72. i += 5;
73. continue;
74. }
75. } else if (c>>1 == six\_byte) {
76. k = (unsigned char)str[i+1];
77. m = (unsigned char)str[i+2];
78. n = (unsigned char)str[i+3];
79. p = (unsigned char)str[i+4];
80. q = (unsigned char)str[i+5];
81. if ( is\_utf8\_special\_byte(k)
82. && is\_utf8\_special\_byte(m)
83. && is\_utf8\_special\_byte(n)
84. && is\_utf8\_special\_byte(p)
85. && is\_utf8\_special\_byte(q) ) {
86. utf8\_yes++;
87. i += 6;
88. continue;
89. }
90. }
91. utf8\_no++;
92. i++;
93. }
94. printf("%d %d\n", utf8\_yes, utf8\_no);
95. int ret = (100\*utf8\_yes)/(utf8\_yes + utf8\_no);
96. if (ret > 90) {
97. return 1;
98. } else {
99. return 0;
100. }
101. }

实现原理：判断网页文本中符合utf-8规则的字数和不符合utf-8规则的字数  
如果符合的字数超过90%，则判断为utf-8编码  
  
其次应该是gb2312编码的判断，由于gb2312相对gbk和big5的编码范围要小，所以  
在gb2312和gbk和big5之间，应该首先判断该网页文本是否是gb2312  
函数实现：

**点击(此处)折叠或打开**

1. int Encoder::is\_gb2312\_code(const string& str)
2. {
3. unsigned one\_byte = 0X00; //binary 00000000
4. int gb2312\_yes = 0;
5. int gb2312\_no = 0;
6. unsigned char k = 0;
7. unsigned char c = 0;
8. for (uint i=0; i<str.size();) {
9. c = (unsigned char)str[i];
10. if (c>>7 == one\_byte) {
11. i++;
12. continue;
13. } else if (c >= 0XA1 && c <= 0XF7) {
14. k = (unsigned char)str[i+1];
15. if (k >= 0XA1 && k <= 0XFE) {
16. gb2312\_yes++;
17. i += 2;
18. continue;
19. }
20. }
22. gb2312\_no++;
23. i += 2;
24. }
25. printf("%d %d\n", gb2312\_yes, gb2312\_no);
26. int ret = (100\*gb2312\_yes)/(gb2312\_yes+gb2312\_no);
27. if (ret > 90) {
28. return 1;
29. } else {
30. return 0;
31. }
32. }

实现原理：统计符合gb2312编码特征的字数和不符合gb2312编码特征的字数  
如果符合的字数超过90%，则判断该网页文本为gb2312  
  
再者应该判断big5编码，原因是因为gbk的范围要比big5的范围广  
函数实现

**点击(此处)折叠或打开**

1. int Encoder::is\_big5\_code(const string& str)
2. {
3. unsigned one\_byte = 0X00; //binary 00000000
4. int big5\_yes = 0;
5. int big5\_no = 0;
6. unsigned char k = 0;
7. unsigned char c = 0;
8. for (uint i=0; i<str.size();) {
9. c = (unsigned char)str[i];
10. if (c>>7 == one\_byte) {
11. i++;
12. continue;
13. } else if (c >= 0XA1 && c <= 0XF9) {
14. k = (unsigned char)str[i+1];
15. if ( k >= 0X40 && k <= 0X7E
16. || k >= 0XA1 && k <= 0XFE) {
17. big5\_yes++;
18. i += 2;
19. continue;
20. }
21. }
23. big5\_no++;
24. i += 2;
25. }
26. printf("%d %d\n", big5\_yes, big5\_no);
27. int ret = (100\*big5\_yes)/(big5\_yes+big5\_no);
28. if (ret > 90) {
29. return 1;
30. } else {
31. return 0;
32. }
33. }

实现原理同gb2312  
  
最后是gbk的判断  
函数实现

**点击(此处)折叠或打开**

1. int Encoder::is\_gbk\_code(const string& str)
2. {
3. unsigned one\_byte = 0X00; //binary 00000000
4. int gbk\_yes = 0;
5. int gbk\_no = 0;
6. unsigned char k = 0;
7. unsigned char c = 0;
8. for (uint i=0; i<str.size();) {
9. c = (unsigned char)str[i];
10. if (c>>7 == one\_byte) {
11. i++;
12. continue;
13. } else if (c >= 0X81 && c <= 0XFE) {
14. k = (unsigned char)str[i+1];
15. if (k >= 0X40 && k <= 0XFE) {
16. gbk\_yes++;
17. i += 2;
18. continue;
19. }
20. }
22. gbk\_no++;
23. i += 2;
24. }
25. printf("%d %d\n", gbk\_yes, gbk\_no);
26. int ret = (100\*gbk\_yes)/(gbk\_yes+gbk\_no);
27. if (ret > 90) {
28. return 1;
29. } else {
30. return 0;
31. }
32. }

实现原理同gb2312和big5  
3、各代码之间的相互转换  
在linux下可以利用iconv实现转换，参考  
<http://blog.chinaunix.net/uid-29594401-id-4257317.html>