[Code：Blocks 中文乱码问题原因分析和解决方法！](http://blog.csdn.net/softman11/article/details/6121538)

2011-01-07 00:20 6150人阅读 [评论](http://blog.csdn.net/softman11/article/details/6121538#comments)(15) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/softman11/article/details/6121538#report)

[编译器](http://blog.csdn.net/tag/details.html?tag=%e7%bc%96%e8%af%91%e5%99%a8)[encoding](http://blog.csdn.net/tag/details.html?tag=encoding)[windows](http://blog.csdn.net/tag/details.html?tag=windows)[gcc](http://blog.csdn.net/tag/details.html?tag=gcc)[character](http://blog.csdn.net/tag/details.html?tag=character)[java](http://blog.csdn.net/tag/details.html?tag=java)

这里以Code::Blocks 10.5版本为蓝本进行说明。

首先，请在Code::Blocks里面输入标准的一个C程序：

**[cpp]** [view plaincopy](http://blog.csdn.net/softman11/article/details/6121538)

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<wchar.h>
4. #include<string.h>
5. #include<locale.h>
6. **int** main(**void**)
7. {
9. **char** str[]="中国china";
10. **wchar\_t** str\_w[]=L"中国china";
12. **int** len=(**int**)strlen(str);
13. **int** len\_w=(**int**)wcslen(str\_w);
14. printf("%s,size=%d/n",str,len);
16. setlocale(LC\_ALL, "chs");
17. wprintf(L"%s,size=%d/n",str\_w,len\_w);
19. system("pause");
20. **return** 0;
21. }

然后选用不同的编译器，观看效果。

1.Tiny C

   编译没问题，但是wprintf是显示不出内容的。注释掉setlocale(LC\_ALL,"chs“)之后，wprintf显示出和printf一样的效果。

结果证明，Tinny C是不真正的支持wchar\_t宽字符集。因为通过调试器，我发现他的wchar\_t里面保存的字符每个字符确实是用2个字节了，但是里面编码依然是ANSI编码，并不是unicode代码，所以setlocale(LC\_ALL,"chs“)+wprintf显示不出来(其实wprintf实现的时候，是不会直接输出unicode的，他实际是先把unicode转成多字节的ANSI编码，然后再输出，和printf原理一样，就是多了一个转码过程,所以你使用之前必须先设置locale,否则他不知道如何转，就输不出来)。

虽然他自称部分支持C99，但是至少在宽字符方面，支持的一点都不好。

Tinny C有一点好，他没有乱码，他要么不显示，要么正常显示。

2.VC2005-2010

  一切OK，没有乱码。是支持wchar\_t支持的最好的！

3.GCC(MinGW)

很遗憾，全是乱码！和java 一个德行（相信用过Java的人一定会想起Java的乱码解决花费的时间吧）。呵呵。但是GCC是支持wchar\_t的，为什么会这样？其实根本原因就是：本地化做的不好。

但是解决方法是有的。

要解决这个问题，先要搞清楚有三个地方涉及到编码问题。

1.Code：:Blocks 编辑器保存源文件用的编码。

   默认情况下，是保存为windows本地编码的，也就是WINDOWS-936字符集，也就是GBK编码。

  但是很神奇的是，GCC编译器默认编译的时候是按照UTF-8解析的。你存成GBK，但是当成UTF-8解析，这还能编译通过，这才有鬼了，所以这两个地方编码不统一好，编译的时候报错：error: converting to execution character set: Illegal byte sequence，你根本连通过编译的可能性都没有！

其实要解决这个问题很简单，编写Code：：Blocks的人只需要在调用编译器之前检测一下源文件是什么编码，然后就自动让编译器用什么编码进行解释，问题就解决了。只是很可惜，Code::Blocks编写的人可能还没有这么做，或许是对本地化认识不够吧，也可能是觉得没必要吧？（所以就给初学的人带来问题了，所以就觉得易用性不如微软了，免费和商业的东西还是有差距的。。。）

2。GCC编译器编译的时候对输入的源文件解释用的编码

这个编译器可以设置-finput-charset=charset来指定编译器用什么编码解释输入源文件。比如如果源文件的字符集是GBk,那么就必须指定-finput-charset=GBK,如果不指定，一律当做UTF-8处理。

除非你源文件真的是UTF-8，否则就会出现转换错误。

3。编译好的执行文件所用编码

  如果你1和2两个地方的编码都能统一，那么编译时不会报错了，但是编译好了，运行一下看看，在控制台显示的依然是乱码！

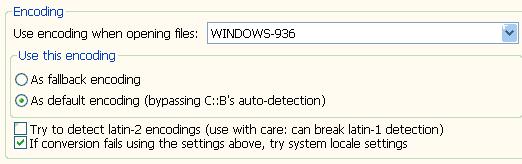
 那是因为控制台显示的时候缺省的是使用系统默认的字符集，比如windows下用的是GBk，但是默认情况下，编译之后的执行文件时编译成UTF-8的，所以又出现了不统一，乱码由此而生！

解决的方法和简单，就是给编译器加上选项：-fexec-charset=GBK，和windows默认的统一，就OK了。

搞懂了乱码产生的原因，那么不难得出结论，如何修改，你想修改成什么都OK，关键是要统一，并不是像网上一些人说的，修改成GBK就OK，其实你要修改成UTF-8都OK，关键是统一。

下面说说修改的地方。

1。修改源文件保存编码在：settings->Editor->gernal settings 看到右边的Encoding group Box了吗？如下图所示：



  Use encoding when opening files:这个表示打开文件用的格式，第一次保存文件的时候也会用这个格式。

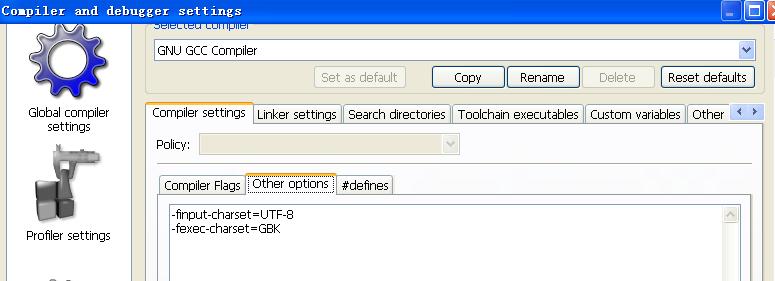
  As default encoding:表示设置为文件缺省保存和打开编码格式

  注意，要先设置好，然后保存文件，才有效。如果你已经保存了文件，无论你怎么修改这个设置，也不会改变你文件的格式了。你的文件还是保持第一次保存的时候的格式。

  所以，如果遇到无法生效，只能先设置好格式，再重新建文件了。

2。修改编译器对源文件解释编码格式和生成执行文件执行时候采用的编码格式

是在settings->compiler and debugger settings里面，选择对应的GCC编译器，如下图所示：



在other options里面加入：

-finput-charset=charset

-fexec-charset=charset

第一个参数表示编译的时候输入文件的编码解释格式，第二参数表示生成的执行文件执行的时候显示用的编码格式。

这些参数如果和实际不吻合，必然产生乱码。只要吻合，就不会乱码了。

由于我的源文件格式是WINDOWS-936,但是这里设置成UTF-8，所以编译肯定报错！

只需要修改成-finput-charset=WINDOWS-936或者GBk，就编译通过了。

如果不设置fexec-charset默认会认为执行环境是UTF-8，而windows下并不是，所以Linux下没问题，因为Linux就是UTF-8的，但是windows 下必然出现乱码。

所以设置成GBk，就统一了。

一切都那么简单，其实，只是因为编程的人做的不够完善，所以才会给使用的人带来困扰。希望这篇文章能帮到一些初学者。或者遇到同样问题的人。