问 c++字符串的编码？

c++ c 编码 乱码 **nightwatch** 2016年04月22日提问

c++ 字符串加载到内存里面是什么编码格式的？

win7中文系统下，控制台默认是GBK编码的，用GBK格式保存的源文件，中文字符串在vs2010下编译输出到控制台会正常输出

但是vs2010里面采用utf-8无BOM的源文件 输出中文字符串到终端就出现乱码了

所以 是不是c++把字符串的值加载到内存中时 是按照cpp文件的编码保存的？ 也就是说utf-8编码的cpp文件，编译后字符串加载到内存时是utf-8编码的？

2016年04月22日提问 1 评论 邀请回答 编辑 更多

1个回答

默认排序 时间排序

已采纳

这个问题想要说清楚还是挺复杂的，题主可以参考 New Options for Managing Character Sets in the Microsoft C/C++ Compiler 这篇文章。具体来说，源代码文件中的字符串常量能否正确的显示在控制台窗口中与以下几个因素有关：

- 源代码文件（.cpp）保存时所使用的字符集（下文用 C1 代替）
- 编译器在读取源代码文件时所使用的内部字符集（source character set）（下文用 C2 代替）
- 编译器在生编译时所使用的字符集（execution character set）（下文用 C3 代替）
- 运行可执行程序的控制台窗口所使用的字符集（下文用 C4 代替）

第一，“源代码文件保存时所使用的字符集”决定了源代码文件中的**抽象字符**保存到硬盘中是什么样的**字节**。这里所说的抽象字符，就是指人所能识别的文字（如“张”）。而字节则是由抽象字符按照字符集的规定映射的，不同的字符集映射的结果不一样（如“张”在 UTF-8 下映射的字节是 `E5 BC A0` 三个字节，而在 GBK 下映射的字节是 `D5 C5` 两个字节）。比如下面一行代码：

```
char *s = "张三";
```

其中的字符串常量 `"张三"` 在使用不同的字符集（C1）保存时，映射的字节是不同的。

第二，“编译器在读取源代码文件时所使用的内部字符集”决定了编译器如何把读入的源代码文件字节流进行**转换**。我所谓的转换就是指把字节流从一种编码转到另一种编码。举例来说，对于抽象字符“张”，如果采用 GBK 编码，映射的字节流就是 `D5 C5` ，而转换到 UTF-8 编码，就是把这个字节流转换成 `E5 BC A0` 。

这个字符集（C2）是由编译器决定的，标准中并未规定。不同的编译器可能采用不同的内部字符集，同样的编译器不同版本也可能采用不同的内部字符集。在 Visual C++ 的某些版本中，内部字符集是 UTF-8，编译器会尝试判断源文件所使用的字符集（C1），并将其转换成内部字符集（C2），**如果编译器不能判断出文件所使用的字符集，则会默认其（C1）为当前操作系统的代码页（default code page）**（*这里如果判断错误，就会造成乱码或编译出错*）。

比如：源文件如果是 UTF-8 with BOM，则能够正确的被 Visual C++ 识别，其中的抽象字符“张”映射的字节流 `E5 BC A0` 就会被正确的转换成 `E5 BC A0`（没变）。而源文件如果是 UTF-8 without BOM，则不能正确的被 Visual C++ 识别，编译器会采用当前的代码页来进行转换，其中的抽象字符“张”映射的字节流 `E5 BC A0` 就会被当作 GBK 编码，错误的转换成其它字节流 `E5 AF AE ...`（寮）。

第三，“编译器在编译时所使用的字符集”决定了编译器如何把源代码中使用内部字符集（C2）编码的字符/字符串常量转还到编译时所使用的字符集（C3）。这个字符集（C3）也是由编译器决定的，标准中并未规定。C++ 中的字符常量和字符串常量有不同的类型，它们对应于不同的 C3。

在 Visual C++ 中，参考 String and Character Literals 和前文提到的博客，可以推知不同类型的字符/字符串常量对应的 C3：

```
// Character literals
auto c0 =  'A'; // char, encoded as default code page
auto c1 = u8'A'; // char, encoded as UTF-8
auto c2 =  L'A'; // wchar_t, encoded as UTF-16LE
auto c3 =  u'A'; // char16_t, encoded as UTF-16LE
auto c4 =  U'A'; // char32_t, encoded as UTF-32LE

// String literals
auto s0 =  "hello"; // const char*, encoded as default code page
auto s1 = u8"hello"; // const char*, encoded as UTF-8
auto s2 =  L"hello"; // const wchar_t*, encoded as UTF-16LE
auto s3 =  u"hello"; // const char16_t*, encoded as UTF-16LE
auto s4 =  U"hello"; // const char32_t*, encoded as UTF-32LE
```

编译器根据字符串常量的类型把其从 C2 转换到 C3（*前面如果判断错误，这里就会继续保留错误*）。

比如： `auto s1 = "张";` ，抽象字符“张”在 C2 中（UTF-8）映射的字节流 `E5 BC A0` 就会被转换成在 C3 中（CP936，GBK）映射的字节流 `D5 C5` 。

`auto s2 = u8"张";` ，抽象字符“张”在 C2 中（UTF-8）映射的字节流 `E5 BC A0` 就会被转换成在 C3 中（UTF-8）映射的字节流 `E5 BC A0` （不变）。

第四，“运行可执行程序的控制台窗口所使用的字符集”决定了如何把编译好的可执行程序中的字节流转换成抽象字符显示在控制台中。比如，在上一步中的 `s1` 映射的字节流就会通过 C4（CP 936）映射回抽象字符“张”，在我们看来就是正确的。而上一步中的 `s2` 映射的字节流就会通过 C4（CP 936）映射回抽象字符“寮”，在我们看来就是乱码。

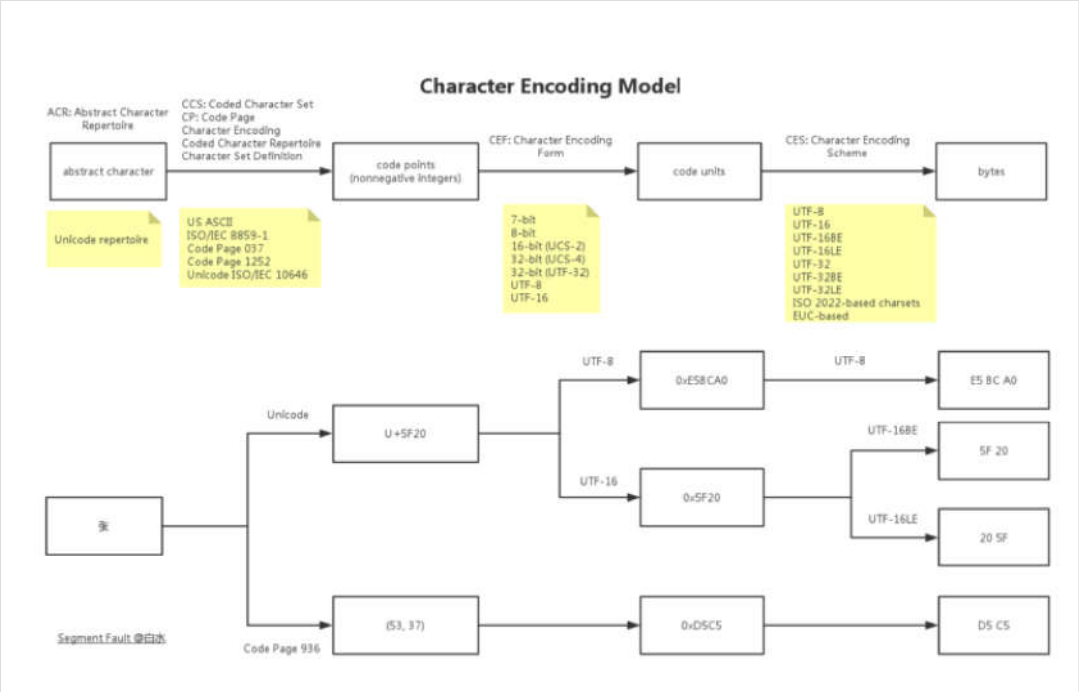
以上，就是我理解的 C++ 中字符/字符串的编码处理方式，如果有误还请指出:-)

题主可以尝试在 Visual C++ 中把以下代码分别保存成 CP936、UTF-8 with BOM、UTF-8 without BOM 的格式，看看输出结果是什么。

```
using namespace std;

int main() {
  char *s1 = u8"张";
  char *s2 = "张";
  cout << "s1 " << sizeof(s1) << " " << strlen(s1) << " -> " << s1 << endl;  // Error in console
  cout << "s2 " << sizeof(s2) << " " << strlen(s2) << " -> " << s2 << endl;  // OK in console

  ofstream os("s1.txt");
  if (os.is_open()) {
    os << "s1 " << sizeof(s1) << " " << strlen(s1) << " -> " << s1 << endl;
    os.close();
  }
  ofstream os2("s2.txt");
  if (os2.is_open()) {
    os2 << "s2 " << sizeof(s2) << " " << strlen(s2) << " -> " << s2 << endl;
    os2.close();
  }
  ofstream os3("s3.txt");
  if (os3.is_open()) {
    os3 << "s1 " << sizeof(s1) << " " << strlen(s1) << " -> " << s1 << endl;
    os3 << "s2 " << sizeof(s2) << " " << strlen(s2) << " -> " << s2 << endl;
    os3.close();
  }

  cin.get();
  return 0;
}
```

输出的三个文件中，前两个文件 `s1.txt` 和 `s2.txt` 都能够被正常的文本编辑器猜出其编码格式，从而正确显示内容，但是第三个文件 `s3.txt` 会显示成部分或全部乱码，因为其中既包含了 UTF-8 编码的字节流又包含了 GBK 编码的字节流，所以文本编辑器就不知道该用什么编码来把字节流映射回抽象文本了。

Character Encoding Model



标准引用和参考文档

> From ISO C++11 § 2.3/1
> The *basic source character set* consists of 96 characters: the space character, the control characters representing horizontal tab, vertical tab, form feed, and new-line, plus the following 91 graphical characters:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
_ { } [ ] # ( ) < > % : ; . ? * + - / ^ & | ~ ! = , \ " '
```

From ISO C++11 § 2.3/2
The *universal-character-name* construct provides a way to name other characters.

```
hex-quad:
    hexadecimal-digit hexadecimal-digit hexadecimal-digit hexadecimal-digit
universal-character-name:
    \u hex-quad
    \U hex-quad hex-quad
```

The character designated by the universal-character-name `\UNNNNNNNN` is that character whose character short name in ISO/IEC 10646 is `NNNNNNNN`; the character designated by the universal-character-name `\uNNNN` is that character whose character short name in ISO/IEC 10646 is `0000NNNN`. ...

From ISO C++11 § 2.3/3
The *basic execution character set* and the *basic execution wide-character set* shall each contain all the members of the basic source character set, plus control characters representing alert, backspace, and carriage return, plus a null character (respectively, null wide character), whose representation has all zero bits.
...
The *execution character set* and the *execution wide-character set* are implementation-defined supersets of the basic execution character set and the basic execution wide-character set, respectively. The values of the members of the execution character sets and the sets of additional members are locale-specific.

From ISO C++11 § 2.2/1The precedence among the syntax rules of translation is specified by the following phases.

1. Physical source file characters are mapped, in an implementation-defined manner, to the basic source character set (introducing new-line characters for end-of-line indicators) if necessary. The set of physical source file characters accepted is implementation-defined. ... Any source file character not in the basic source character set (2.3) is replaced by the universal-character-name that designates that character. ...
2. ...
3. ...
4. ...
5. Each source character set member in a character literal or a string literal, as well as each escape sequence and universal-character-name in a character literal or a non-raw string literal, is converted to the corresponding member of the execution character set (2.14.3, 2.14.5); if there is no corresponding member, it is converted to an implementation-defined member other than the null (wide) character.
6. ...

From ISO C++11 § 2.14.3/5
A universal-character-name is translated to the encoding, in the appropriate execution character set, of the character named. If there is no such encoding, the universal-character-name is translated to an implementationdefined encoding. [*Note:* In translation phase 1, a universal-character-name is introduced whenever an actual extended character is encountered in the source text. Therefore, all extended characters are described in terms of universal-character-names. However, the actual compiler implementation may use its own native character set, so long as the same results are obtained. — *end note*]

From ISO C++11 § 2.14.5/6 7 15
6 After translation phase 6, a string literal that does not begin with an encoding-prefix is an ordinary string literal, and is initialized with the given characters.
7 A string literal that begins with u8, such as `u8"asdf"`, is a UTF-8 string literal and is initialized with the given characters as encoded in UTF-8.
...
15 Escape sequences and universal-character-names in non-raw string literals have the same meaning as in character literals (2.14.3), except that the single quote `'` is representable either by itself or by the escape sequence `\'`, and the double quote `"` shall be preceded by a `\`. In a narrow string literal, a universal-character-name may map to more than one char element due to *multibyte encoding*. ...

- Character Sets
- String and Character Literals
- New Options for Managing Character Sets in the Microsoft C/C++ Compiler

2016年04月26日回答 评论 编辑

撰写答案

撰写答案...

讲堂推荐 更多

本周五


Yii2之RESTful程序基础设计及目录规划
kumfo | 12 人参与

2017-04-24 周一


性感的PHP——现代化的PHP开发
纸牌屋弗兰克 | 54 人参与

2017-05-10 周三


□□学习 Vue 你需要知道的 webpack 知识
KingMario | 9 人参与

2017-05-13 周六


Docker：新时代程序猿不可或缺的技术
有明 | 21 人参与

2017-06-04 周日


web 安全之加密与解密技术
耗子 | 41 人参与

相似问题

windows7下c++字符串是gbk编码?还是unicode? 2 回答 | 已解决

chardet.detect无法获取字符编码，字符编码为乱码怎么办？

前端js json字符串base64_encode编码,提交后端中文乱码 1 回答

Android Studio编码格式及运行时中文乱码 1 回答 | 已解决

c++ 读取txt文档后生成霍夫曼编码报错，求指教！ 1 回答 | 已解决

JNI 字符串处理乱码问题jstring to char 2 回答

js ajax 取回数据乱码 2 回答 | 已解决

c++ post django 中文编码问题

为何我C++中'\0'在控制台输出的时候变成a了？ 3 回答

请问PHP中怎么将UTF-8字符串转化为ANSI编码 1 回答 | 已解决

分享扩散：