Subject: LoadFile problem with accented chars
Posted by koldo on Sat, 07 Feb 2009 21:27:36 GMT
View Forum Message <> Reply to Message

Hello all

In a program I am reading text files written with Notepad.

To do that I simply use String LoadFile(String fileName) and I parse through the readed String.

When the file readed has only simple characters everything goes right, but when I enter in

case a simple char \r is converted into a -17 int.

I have seen that TheIde handles this ok. It seems that (it is just a guess) TheIde detects the file charset and handles the file properly converting it to an Utf8 String.

I have tried to get parts of TheIde code into my program but without success. Do you know what to do?.

Best regards
Koldo

Subject: Re: LoadFile problem with accented chars
Posted by mirek on Sun, 08 Feb 2009 07:06:41 GMT
View Forum Message <> Reply to Message

U++ never does any encoding conversions in Stream *content* (it can do some conversions to *file name*, e.g. converting utf-8 to unicode).

I suspect that the error is in processing the file. Hard to say what the problem is without knowing more.

Mirek

Subject: Re: LoadFile problem with accented chars
Posted by koldo on Sun, 08 Feb 2009 21:11:52 GMT
View Forum Message <> Reply to Message

Hello luzr

It seems it is a matter of Notepad itself. If the file has 7 bits chars there is no problem, but after

Using this test program:

```
CONSOLE_APP_MAIN
{
 String data = LoadFile("C:\\test.txt");
 for (int i = 0; i < data.GetCount(); ++i)
  puts(Format("%d: %d", i, data[i]));
 getchar();
}
```

```
0: 97
1: 45
2: -31
```

but after saving and opening the file some times, I get this:

```
0: -1
1: -2
2: 97
3: 0
4: 45
5: 0
6: -31
7: 0
```

and yesterday I got other output... The answer is that Notepad adds a "BOM" to the file if it thinks it requires a bigger encoding.

BOM (Byte Order Mark, http://unicode.org/faq/utf_bom.html#BOM) is a signature of letters in the begining of files that shows its encoding. For example:

- EF BB BF means UTF-8
- FF FE means UTF-16, little-endian

So yesterday Notepad saved the file as UTF8 (beginning with -17 == EF) and today it saved it in UTF-16, little-endian (beginning with a -1 == FF)

Sorry, perhaps it is not easy but, do you know how to program a way to get a text file and converting it into utf8 to be properly viewed U++ programs?, as when entering these chars into U++ controls I get strange symbols and errors. It will also be great for parsing them.

Best regards
Koldo

Subject: Re: LoadFile problem with accented chars
Posted by mirek on Mon, 09 Feb 2009 07:12:01 GMT
View Forum Message <> Reply to Message

koldo wrote on Sun, 08 February 2009 16:11Hello luzr

It seems it is a matter of Notepad itself. If the file has 7 bits chars there is no problem, but after

Using this test program:

```
CONSOLE_APP_MAIN
{
 String data = LoadFile("C:\\test.txt");
 for (int i = 0; i < data.GetCount(); ++i)
  puts(Format("%d: %d", i, data[i]));
 getchar();
}
```

```
0: 97
1: 45
2: -31
```

but after saving and opening the file some times, I get this:

```
0: -1
1: -2
2: 97
3: 0
4: 45
5: 0
6: -31
7: 0
```

and yesterday I got other output... The answer is that Notepad adds a "BOM" to the file if it thinks it requires a bigger encoding.

BOM (Byte Order Mark, http://unicode.org/faq/utf_bom.html#BOM) is a signature of letters in the begining of files that shows its encoding. For example:

- EF BB BF means UTF-8
- FF FE means UTF-16, little-endian

Why do not interpret it yourself?

I suggest implementing these:

```
WString LoadBOMW(const Stream& s);
WString LoadFileBOMW(const char *path);
void    SaveBOMUtf8(const Stream& s, const WString& data);
bool    SaveFileBOMUtf8(const char *path, const WString& data);

String  LoadBOM(const Stream& s); // Default encoding, usually utf-8
String  LoadFileBOM(const char *path);
void    SaveBOMUtf8(const Stream& s, const String& data);
bool    SaveFileBOMUtf8(const char *path, const String& data);
```

I would be glad to add them to Core.

Mirek

Subject: Re: LoadFile problem with accented chars
Posted by koldo on Mon, 09 Feb 2009 07:47:21 GMT
View Forum Message <> Reply to Message

Hello luzr

No problem. One question: Is there inside U++ code to convert a string from/to a certain charset? (for not trying to reinvent the wheel). Inside there are functions with names as pretty as:

ToUnicode
FromUnicode
ConvertCharset
CheckUtf8
FromUtf8
ToUtf8

Best regards
Koldo

Subject: Re: LoadFile problem with accented chars
Posted by mirek on Mon, 09 Feb 2009 16:28:16 GMT
View Forum Message <> Reply to Message

koldo wrote on Mon, 09 February 2009 02:47Hello luzr

No problem. One question: Is there inside U++ code to convert a string from/to a certain charset? (for not trying to reinvent the wheel). Inside there are functions with names as pretty as:

ToUnicode

FromUnicode
ConvertCharset
CheckUtf8
FromUtf8
ToUtf8

Best regards
Koldo

Sure!

Core/Charset.h

They work

Mirek

Subject: Re: LoadFile problem with accented chars
Posted by koldo on Tue, 10 Feb 2009 08:23:50 GMT
View Forum Message <> Reply to Message

Hello luzr

I have had problems. In reading an UTF16-Little Endian, ToUtf8 does not seem to do it well.

I have used this function:

```
String  LoadFileBOM(const char *path)
{
 String s = LoadFile(path);
 if (((s[0]&0xFF) == 0xFF) && ((s[1]&0xFF) == 0xFE))     // UTF16 Little Endian
  s = ToUtf8(s.Mid(2).ToWString());
 else if (((s[0]&0xFF) == 0xEF) && ((s[1]&0xFF) == 0xBB) && ((s[2]&0xFF) == 0xBF)) // UTF8
  s = s.Mid(3);
 return s;
}
```

that is called from here:

```
String s = LoadFileBOM("demo_u_16le.txt");
String ss;

for (int i = 0; i < s.GetCount(); ++i)
 ss.Cat(Format("%d: %0x;\n", i, s[i]&0xFF));

ss.Cat(s);
TestLineEdit.SetData(ss);
TestEditString.SetData(ss);
```

TestDocEdit.SetData(ss);

Quote:0: 41;
1: 75;
2: 70;
3: c3;
4: a1;

but it is:

Quote:0: 41;
1: 0;
2: 75;
3: 0;
4: 70;
5: 0;
6: e1;
7: 0;

Another question. When loading an UTF8 file with the later code, LineEdit and DocEdit reads it right but EditString shows a strange char. Does EditString not show UTF8 or I am doing it wrong?.

Best regards
Koldo

Subject: Re: LoadFile problem with accented chars
Posted by koldo on Wed, 11 Feb 2009 14:05:52 GMT
View Forum Message <> Reply to Message

Hello luzr

Everything solved. I inclose you some of the proposed functions:

```
String  LoadFileBOM(const char *path)
{
 String s = LoadFile(path);
 if (((s[0]&0xFF) == 0xFF) && ((s[1]&0xFF) == 0xFE)) {   // UTF16 Little Endian
  StringBuffer ws = s.Mid(2);
  s = ToUtf8((wchar *)ws.Begin(), ws.GetCount()*sizeof(char)/sizeof(wchar));
 } else if (((s[0]&0xFF) == 0xEF) && ((s[1]&0xFF) == 0xBB) && ((s[2]&0xFF) == 0xBF)) // UTF8
  s = s.Mid(3);
 else          // May be ISO8859-1
  s = ToUtf8(ToUnicode(s, CHARSET_ISO8859_1));
```

```
 return s;
}
bool SaveBOMUtf8(Stream& out, const String& data) {
 if(!out.IsOpen() || out.IsError())
  return false;
 unsigned char bom[] = {0xEF, 0xBB, 0xBF};
 out.Put(bom, 3);
 out.Put((const char *)data, data.GetLength());
 out.Close();
 return out.IsOK();
}
bool SaveFileBOMUtf8(const char *path, const String& data)
{
 FileOut out(path);
 return SaveBOMUtf8(out, data);
}
```

When loading it checks the BOM if it is UTF-16 little endian or UTF-8. If there is no BOM it is considered to be ISO8859-1. It always return a UTF-8 String.

When saving it always save to UTF-8.

If they are right I will do the rest of functions.

There was no problem with EditString. My error was because it handles UTF-8 but not ISO8859-1 chars.

Best regards
Koldo

Subject: Re: LoadFile problem with accented chars
Posted by cbpporter on Wed, 11 Feb 2009 14:25:08 GMT
View Forum Message <> Reply to Message

s = ToUtf8(s.Mid(2).ToWString());
Well by looking at this code from the old version, I believe you should be getting the results you were getting (the wrong ones).

When calling ToWString, the input will be considered to be in Utf8, but you have an Utf16 stuffed inside an String (Utf8). That explains the extra zeros.

On the other hand, the new versions may work, but they are a little heavy on conversions and allocations so I wouldn't use them on large files. Especially for Utf8 BOM, I believe the solution would be to create a LoadFile which detects the BOM and allocates and fills a buffer/String without the BOM directly. I'll have to check out LoadFile before I can elaborate on this.

Subject: Re: LoadFile problem with accented chars
Posted by koldo on Wed, 11 Feb 2009 18:26:06 GMT
View Forum Message <> Reply to Message

Sorry cbpporter

In some hours I will pass a more optimized proposal.

Best regards
Koldo

Subject: Re: LoadFile problem with accented chars
Posted by koldo on Thu, 12 Feb 2009 00:13:52 GMT
View Forum Message <> Reply to Message

Hello luzr and all

Here I inclose you the "String" version of the functions.

LoadStreamBOM now handles UTF-16 LE & BE, UTF-8 and ISO8859_1 text files and is more optimized but more complex than the first version.

Best regards
Koldo

```
String LoadStreamBOM(Stream& in)
{
 if(in.IsOpen()) {
  in.ClearError();
  int size = (int)in.GetLeft();
  if((dword)size != 0xffffffff) {
   unsigned char header[3];        // Get 3 bytes header
   if (!in.GetAll(&header, 3))
    return String::GetVoid();
   if ((header[0] == 0xFF) && (header[1] == 0xFE)) {  // Check header
    StringBuffer s(size-2);        // UTF16 Little Endian
    s[0] = header[2];         // This char is not header
    if (!in.GetAll(s.Begin()+1, size-3))
     return String::GetVoid();      // Conversion
    return ToUtf8((wchar *)s.Begin(), (size-2)*sizeof(char)/sizeof(wchar));
   } else if ((header[0] == 0xFE) && (header[1] == 0xFF)) {
    StringBuffer s(size-2);        // UTF16 Big Endian
    s[0] = header[2];         // This char is not header
    if (!in.GetAll(s.Begin()+1, size-3))
     return String::GetVoid();
    for (int i = 0; i < size-2; i += 2) { // Change from big to little endian
```

```
     unsigned char aux = s[i];   // by changing byte order
     s[i] = s[i+1];
     s[i+1] = aux;
    }            // Conversion
    return ToUtf8((wchar *)s.Begin(), (size-2)*sizeof(char)/sizeof(wchar));
   } else if ((header[0] == 0xEF) && (header[1] == 0xBB) && (header[2] == 0xBF))
    return in.Get(size-3);        // UTF8. No conversion required
   else {
    StringBuffer s(size);        // Maybe ISO8859-1
    s[0] = header[0];         // Three chars are not header
    s[1] = header[1];         // so inserted into the StringBuffer
    s[2] = header[2];
    if (!in.GetAll(s.Begin()+3, size-3))
     return String::GetVoid();
    return ToUtf8(ToUnicode(s.Begin(), size, CHARSET_ISO8859_1)); // Conversion
   }
  }
 }
 return String::GetVoid();
}
String LoadFileBOM(const char *filename)
{
 FileIn in(filename);
 return LoadStreamBOM(in);
}
bool SaveBOMUtf8(Stream& out, const String& data) {
 if(!out.IsOpen() || out.IsError())
  return false;
 unsigned char bom[] = {0xEF, 0xBB, 0xBF};
 out.Put(bom, 3);
 out.Put((const char *)data, data.GetLength());
 out.Close();
 return out.IsOK();
}
bool SaveFileBOMUtf8(const char *path, const String& data)
{
 FileOut out(path);
 return SaveBOMUtf8(out, data);
}
```

Subject: Re: LoadFile problem with accented chars
Posted by mirek on Thu, 12 Feb 2009 17:02:08 GMT
View Forum Message <> Reply to Message

I am afraid this is not really consistent with the rest of U++ handling of charsets.

The problem is that you always convert to Utf8. I think we should rather convert to active default

encoding (which usually IS UTF-8, but this is how things work and in fact, some of my application depend on it).

I also think the we should have 'W' variant (returing WString) first, then 'String' variant with conversion - that will cost nothing....

Mirek

Subject: Re: LoadFile problem with accented chars
Posted by koldo on Fri, 13 Feb 2009 08:50:30 GMT
View Forum Message <> Reply to Message

Hello luzr

No problem.

I have done it and I have prepared a simple demo test saving text files with different codifications and loading them into EditString, LineEdit and DocEdit controls.

It is tested in XP (MinGW and MSC). This afternoon I will test in GNU/Linux and I will post it.

Best regards
Koldo

(I do not know Czech so I hope this text is not inadecuate)

File Attachments
1) Screen.JPG, downloaded 637 times

Subject: Re: LoadFile problem with accented chars
Posted by mirek on Fri, 13 Feb 2009 10:05:05 GMT
View Forum Message <> Reply to Message

Well, before reading your post, I might have duplicated some of your recent efforts - based on your code, I have added BOM support to Core.... (Core/Bom.cpp).

I hope there is a couple additional problems solved, e.g. system charset is used if no BOM header is detected, String and StringW share single function body while avoiding unnecessarry conversion (utf8->wstring->utf8).

Mirek

Subject: Re: LoadFile problem with accented chars
Posted by koldo on Fri, 13 Feb 2009 18:09:45 GMT
View Forum Message <> Reply to Message

Hello luzr

After testing it in Linux here I enclose you the code with the demo.

Thinking about the functions name perhaps it is not the best as not many people knows that many apparently plain text files have this BOM.

Perhaps managing the BOM would have to be the by default behaviour of LoadFile and SaveFile functions and bypassing the BOM would be only an option.

Best regards
Koldo

File Attachments
1) DemoBOM.7z, downloaded 193 times

Subject: Re: LoadFile problem with accented chars
Posted by mirek on Sat, 14 Feb 2009 23:05:22 GMT
View Forum Message <> Reply to Message

koldo wrote on Fri, 13 February 2009 13:09
Perhaps managing the BOM would have to be the by default behaviour of LoadFile and SaveFile functions and bypassing the BOM would be only an option.

LoadFile / SaveFile must be able to load normal binary files. BOM is relatively high-end for them.

Mirek