Subject: BUG: Serious problem with Core/CharSet
Posted by kov_serg on Thu, 03 Jun 2010 22:22:23 GMT
View Forum Message <> Reply to Message

Upp2361/WinXP

Testing code shows "error 128=31"

```
#include <Core/Core.h>

using namespace Upp;
CONSOLE_APP_MAIN {
 wchar_t w[]={0x0410,0}; // must be converted to 0x80,0 see definition of CHRTAB_CP866
 String r=FromUnicode(w,CHARSET_CP866);
 int rc=r[0]&255;
 Cout()<<((rc==0x80)?"ok ":"error 128=")<<rc<<"\n";
}
```

CHARSET_CP866 has code 62 but if we use 60 instead 62 it will works.

The bug hides here:

Core\CharSet.cpp:2252

```
...
byte  ResolveCharset(byte charset)
{
 return charset ? charset : DefaultCharset;
}

inline
static CharSetData& s_cset(byte charset)
{
 return s_map()[ResolveCharset(charset)]; // <<<< why charset code used here instead of real
index String("CP866") ???
}
```

s_map() returns ArrayMap<String, CharSetData>& and shuld be indexed by string "CP866" not by charset code 62. Because it real index is 60.

Look at Core\CharSet.cpp:2206

```
 int q = s_map().GetCount();
```

And try to ASSERT(q==systemcharset); and this assert fail almost always.

May be at beginig charset codes was equal to s_map item index, but now this is not true. And this

is a problem function FromUnicode works wrong.

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by tojocky on Fri, 04 Jun 2010 05:28:41 GMT
View Forum Message <> Reply to Message

kov_serg wrote on Fri, 04 June 2010 01:22Upp2361/WinXP

Testing code shows "error 128=31"

```
#include <Core/Core.h>

using namespace Upp;
CONSOLE_APP_MAIN {
 wchar_t w[]={0x0410,0}; // must be converted to 0x80,0 see definition of CHRTAB_CP866
 String r=FromUnicode(w,CHARSET_CP866);
 int rc=r[0]&255;
 Cout()<<((rc==0x80)?"ok ":"error 128=")<<rc<<"\n";
}
```

CHARSET_CP866 has code 62 but if we use 60 instead 62 it will works.

The bug hides here:

Core\CharSet.cpp:2252

```
...
byte  ResolveCharset(byte charset)
{
 return charset ? charset : DefaultCharset;
}

inline
static CharSetData& s_cset(byte charset)
{
 return s_map()[ResolveCharset(charset)]; // <<<< why charset code used here instead of real
index String("CP866") ???
}
```

s_map() returns ArrayMap<String, CharSetData>& and shuld be indexed by string "CP866" not by charset code 62. Because it real index is 60.

Look at Core\CharSet.cpp:2206

```
 int q = s_map().GetCount();
```

And try to ASSERT(q==systemcharset); and this assert fail almost always.

May be at beginig charset codes was equal to s_map item index, but now this is not true. And this is a problem function FromUnicode works wrong.

Some days ago I had problems with charset. but the problem was not with u++. u++ works perfectly. the chars tables and conversions are perfectly.

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by Mindtraveller on Fri, 04 Jun 2010 06:02:10 GMT
View Forum Message <> Reply to Message

There is definitely something wrong with charset conversion:

```
CONSOLE_APP_MAIN
{
 Cout() << "\n";

 for (int i=0; i<s.GetLength(); ++i)
  Cout() << Format("%02X ", s[i]);
}
```

gives output:
1F 1F 1F 1F 20 1F 1F 1F 1F 1F 1F 21
which means no Russian characters are actually converted.

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by kov_serg on Fri, 04 Jun 2010 06:33:59 GMT
View Forum Message <> Reply to Message

The poblem exist! Just belive me and try to test yourself.
Cheking code:

```
#include <Core/Core.h>
using namespace Upp;
CONSOLE_APP_MAIN {
 for(int i=0;i<256;++i) {
  String name=CharsetName(i);
  if (name.GetLength()) {
   while(name.GetLength()<18) name+=" ";
   Cout()<<"#define  CHARSET_"<<name<<"\t"<<i<<"\n";
  }
 }
}
```

Then look at defines in CharSet.h

Here is the difference

```
// in Core/CharSet.h                    // real case
#define  CHARSET_CP853           51      // #define  CHARSET_CP853           49
#define  CHARSET_CP855           52      // #define  CHARSET_CP855           50
#define  CHARSET_CP856           53      // #define  CHARSET_CP856           51
#define  CHARSET_CP857           54      // #define  CHARSET_CP857           52
#define  CHARSET_CP858           55      // #define  CHARSET_CP858           53
#define  CHARSET_CP860           56      // #define  CHARSET_CP860           54
#define  CHARSET_CP861           57      // #define  CHARSET_CP861           55
#define  CHARSET_CP862           58      // #define  CHARSET_CP862           56
#define  CHARSET_CP863           59      // #define  CHARSET_CP863           57
#define  CHARSET_CP864           60      // #define  CHARSET_CP864           58
#define  CHARSET_CP865           61      // #define  CHARSET_CP865           59
#define  CHARSET_CP866           62      // #define  CHARSET_CP866           60
#define  CHARSET_CP869           63      // #define  CHARSET_CP869           61
#define  CHARSET_CP874           64      // #define  CHARSET_CP874           62
#define  CHARSET_CP922           65      // #define  CHARSET_CP922           63
#define  CHARSET_GEORGIAN_ACADEMY  66      // #define  CHARSET_Georgian-Academy
64
#define  CHARSET_GEORGIAN_PS      67      // #define  CHARSET_Georgian-PS       65
#define  CHARSET_HP_ROMAN8        68      // #define  CHARSET_HP-ROMAN8         66
#define  CHARSET_ISO_8859_1       69      // #define  CHARSET_iso8859-1         1
#define  CHARSET_ISO_8859_10      70      // #define  CHARSET_iso8859-10        10
#define  CHARSET_ISO_8859_13      71      // #define  CHARSET_iso8859-13        11
#define  CHARSET_ISO_8859_14      72      // #define  CHARSET_iso8859-14        12
#define  CHARSET_ISO_8859_15      73      // #define  CHARSET_iso8859-15        13
#define  CHARSET_ISO_8859_16      74      // #define  CHARSET_iso8859-16        14
#define  CHARSET_ISO_8859_2       75      // #define  CHARSET_iso8859-2         2
#define  CHARSET_ISO_8859_3       76      // #define  CHARSET_iso8859-3         3
#define  CHARSET_ISO_8859_4       77      // #define  CHARSET_iso8859-4         4
#define  CHARSET_ISO_8859_5       78      // #define  CHARSET_iso8859-5         5
#define  CHARSET_ISO_8859_6       79      // #define  CHARSET_iso8859-6         6
#define  CHARSET_ISO_8859_7       80      // #define  CHARSET_iso8859-7         7
#define  CHARSET_ISO_8859_8       81      // #define  CHARSET_iso8859-8         8
#define  CHARSET_ISO_8859_9       82      // #define  CHARSET_iso8859-9         9
#define  CHARSET_JIS_X0201        83      // #define  CHARSET_iso8859-10        10
#define  CHARSET_KOI8_RU          85      // #define  CHARSET_KOI8-RU           67
#define  CHARSET_KOI8_T           86      // #define  CHARSET_KOI8-T            68
#define  CHARSET_KOI8_U           87      // #define  CHARSET_KOI8-U            69
#define  CHARSET_MACARABIC        88      // #define  CHARSET_MacArabic         70
#define  CHARSET_MACCENTRALEUROPE  89      // #define  CHARSET_MacCentralEurope
71
#define  CHARSET_MACCROATIAN      90      // #define  CHARSET_MacCroatian       72
#define  CHARSET_MACCYRILLIC      91      // #define  CHARSET_MacCyrillic       73
#define  CHARSET_MACGREEK         92      // #define  CHARSET_MacGreek          74
#define  CHARSET_MACHEBREW        93      // #define  CHARSET_MacHebrew         75
#define  CHARSET_MACICELAND       94      // #define  CHARSET_MacIceland        76
```

```
#define  CHARSET_MACROMAN         95      // #define  CHARSET_MacRoman         77
#define  CHARSET_MACROMANIA       96      // #define  CHARSET_MacRomania       78
#define  CHARSET_MACTHAI          97      // #define  CHARSET_MacThai          79
#define  CHARSET_MACTURKISH       98      // #define  CHARSET_MacTurkish       80
#define  CHARSET_MACUKRAINE       99      // #define  CHARSET_MacUkraine       81
#define  CHARSET_MULELAO_1        100     // #define  CHARSET_MuleLao-1        82
#define  CHARSET_NEXTSTEP         101     // #define  CHARSET_NEXTSTEP         83
#define  CHARSET_RISCOS_LATIN1    102     // #define  CHARSET_RISCOS-LATIN1    84
#define  CHARSET_TCVN             103     // #define  CHARSET_TCVN             85
#define  CHARSET_TIS_620          104     // #define  CHARSET_TIS-620          86
#define  CHARSET_VISCII           105     // #define  CHARSET_VISCII           87
#define  CHARSET_TOASCII          253     // ?????
```

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by mirek on Sat, 05 Jun 2010 19:38:04 GMT
View Forum Message <> Reply to Message

Well, the problem was caused when I have applied a patch adding a lot of charset - but some of them were invalid. So I have removed those invalid, which created "holes" in numbering.

Now it should be fixed (I have filled the holes witch charsets from the end of the list).

Mirek

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by kov_serg on Sat, 05 Jun 2010 20:12:04 GMT
View Forum Message <> Reply to Message

May be ASSERT in "byte AddCharSet(const char *name, const word *table, byte systemcharset)" would help in future

```
...
 int q = s_map().GetCount();
 ASSERT(q>0 && q==systemcharset);
 CharSetData& m = s_map().Add(name);
...
```

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by Mindtraveller on Sun, 06 Jun 2010 10:47:17 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 05 June 2010 23:38Well, the problem was caused when I have applied a patch

adding a lot of charset - but some of them were invalid. So I have removed those invalid, which created "holes" in numbering.

Now it should be fixed (I have filled the holes witch charsets from the end of the list).

Mirek
Looks like my fault. I'll check better next time.

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by Mindtraveller on Mon, 07 Jun 2010 08:24:03 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Fri, 04 June 2010 10:02There is definitely something wrong with charset conversion:

```
CONSOLE_APP_MAIN
{
 Cout() << "\n";

 for (int i=0; i<s.GetLength(); ++i)
  Cout() << Format("%02X ", s[i]);
}
```

gives output:
1F 1F 1F 1F 20 1F 1F 1F 1F 1F 1F 21
which means no Russian characters are actually converted.

The problem is still there.

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by kov_serg on Mon, 07 Jun 2010 10:27:55 GMT
View Forum Message <> Reply to Message

This will work.

```
#undef CHARSET_CP866
#define CHARSET_CP866 60

CONSOLE_APP_MAIN
{
 Cout() << "\n";

 for (int i=0; i<s.GetLength(); ++i)
  Cout() << Format("%02X ", s[i]&255);
}
// outputs: 82 E1 A5 AC 20 AF E0 A8 A2 A5 E2 21
```

Don't panic. I think this problem will be fixed in fututе releases. Now you can fix defines in your CharSet.h

```
...
#define  CHARSET_CP865          59 // instead of 61
#define  CHARSET_CP866          60 // instead of 62
#define  CHARSET_CP869          61 // instead of 63
... and so on
```

Subject: Re: BUG: Serious problem with Core/CharSet
Posted by mirek on Mon, 07 Jun 2010 12:40:14 GMT
View Forum Message <> Reply to Message

Ops. Well, it proved to be tricky to do it this way -> I have heavily refactored the code to make sure numeric IDs are forced to match correctly.

Please check...

Mirek
