Subject: Improved std::[w]string support
Posted by mirek on Wed, 03 Feb 2021 17:02:12 GMT
View Forum Message <> Reply to Message

This is now possible

```
{
 String h = "Hello world";
 std::string sh = h;
 DDUMP(sh);
 h = sh;
 DDUMP(h);
 Value v = sh;
 DDUMP(v);
 sh = v;
 DDUMP(sh);
}

{
 WString h = "Hello world";
 std::wstring sh = h;
 DDUMP(sh);
 h = sh;
 DDUMP(h);
 Value v = sh;
 DDUMP(v);
 sh = v;
 DDUMP(sh);
}
```

...because why not... :)

Subject: Re: Improved std::[w]string support
Posted by Didier on Wed, 03 Feb 2021 17:58:19 GMT
View Forum Message <> Reply to Message

Fantastic :o :o

std::string support in U++ has always been source of problems
This WILL help :)

Subject: Re: Improved std::[w]string support

Posted by mirek on Wed, 03 Feb 2021 18:15:47 GMT
View Forum Message <> Reply to Message

Didier wrote on Wed, 03 February 2021 18:58Fantastic :o :o

std::string support in U++ has always been source of problems
This WILL help :)

Actually, it was always there... Just needed explicit casts here and there. Now it does not.

Subject: Re: Improved std::[w]string support
Posted by Klugier on Wed, 03 Feb 2021 19:07:19 GMT
View Forum Message <> Reply to Message

Hello Mirek,

It seems that this change cause compilation problem on Linux with GCC (a lot of warnings):

```
/home/klugier/upp/uppsrc/Core/Value.h: In function 'bool Upp::operator!=(Upp::String, const
Upp::Value&)':
/home/klugier/upp/uppsrc/Core/Value.h:341:73: error: call of overloaded 'String(const
Upp::Value&)' is ambiguous
  341 | inline bool operator!=(T x, const Value& v)   { return v.Is<VT>() ? (VT)v != x : v != Value(x);
} \
      |                                                                         ^
/home/klugier/upp/uppsrc/Core/Value.h:343:26: note: in expansion of macro
'VALUE_COMPARE_V'
  343 | #define VALUE_COMPARE(T) VALUE_COMPARE_V(T, T)
      |                          ^~~~~~~~~~~~~~~
/home/klugier/upp/uppsrc/Core/Value.h:351:1: note: in expansion of macro 'VALUE_COMPARE'
  351 | VALUE_COMPARE(String)
      | ^~~~~~~~~~~~
In file included from /home/klugier/upp/uppsrc/Core/Core.h:302,
                 from /home/klugier/upp/uppsrc/ide/Builders/BuilderUtils.h:4,
                 from /home/klugier/upp/uppsrc/ide/Builders/BuilderUtils.cpp:1:
/home/klugier/upp/uppsrc/Core/String.h:403:2: note: candidate: 'Upp::String::String(const string&)'
  403 |  String(const std::string& s)                        { String0::Set0(s.c_str(), (int)s.length()); }
      |  ^~~~~~
/home/klugier/upp/uppsrc/Core/String.h:386:2: note: candidate:
'Upp::String::String(Upp::String&&)'
  386 |  String(String&& s)                                  { String0::Pick0(pick(s)); }
      |  ^~~~~~
/home/klugier/upp/uppsrc/Core/String.h:385:2: note: candidate: 'Upp::String::String(const
Upp::String&)'
  385 |  String(const String& s)                             { String0::Set0(s); }
      |  ^~~~~~
```

Klugier

Subject: Re: Improved std::[w]string support
Posted by mirek on Wed, 03 Feb 2021 22:56:40 GMT
View Forum Message <> Reply to Message

Klugier wrote on Wed, 03 February 2021 20:07Hello Mirek,

It seems that this change cause compilation problem on Linux with GCC (a lot of warnings):

```
/home/klugier/upp/uppsrc/Core/Value.h: In function 'bool Upp::operator!=(Upp::String, const
Upp::Value&)':
/home/klugier/upp/uppsrc/Core/Value.h:341:73: error: call of overloaded 'String(const
Upp::Value&)' is ambiguous
  341 | inline bool operator!=(T x, const Value& v)   { return v.Is<VT>() ? (VT)v != x : v != Value(x);
} \
      |                                                                         ^
/home/klugier/upp/uppsrc/Core/Value.h:343:26: note: in expansion of macro
'VALUE_COMPARE_V'
  343 | #define VALUE_COMPARE(T) VALUE_COMPARE_V(T, T)
      |                          ^~~~~~~~~~~~~~~
/home/klugier/upp/uppsrc/Core/Value.h:351:1: note: in expansion of macro 'VALUE_COMPARE'
  351 | VALUE_COMPARE(String)
      | ^~~~~~~~~~~~
In file included from /home/klugier/upp/uppsrc/Core/Core.h:302,
                 from /home/klugier/upp/uppsrc/ide/Builders/BuilderUtils.h:4,
                 from /home/klugier/upp/uppsrc/ide/Builders/BuilderUtils.cpp:1:
/home/klugier/upp/uppsrc/Core/String.h:403:2: note: candidate: 'Upp::String::String(const string&)'
  403 |  String(const std::string& s)                        { String0::Set0(s.c_str(), (int)s.length()); }
      |  ^~~~~~
/home/klugier/upp/uppsrc/Core/String.h:386:2: note: candidate:
'Upp::String::String(Upp::String&&)'
  386 |  String(String&& s)                                  { String0::Pick0(pick(s)); }
      |  ^~~~~~
/home/klugier/upp/uppsrc/Core/String.h:385:2: note: candidate: 'Upp::String::String(const
Upp::String&)'
  385 |  String(const String& s)                             { String0::Set0(s); }
      |  ^~~~~~
```

Klugier

I believe it is a compiler bug and I was not able to navigate around it. Thus above demo had to be changed to

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

using namespace Upp;

CONSOLE_APP_MAIN
{
 StdLogSetup(LOG_COUT|LOG_FILE);
 {
  String h = "Hello world";
  std::string sh = h.ToStd();
  DDUMP(sh);
  h = sh;
  DDUMP(h);
  Value v = sh;
  DDUMP(v);
  sh = v.ToStd();
  DDUMP(sh);
 }

 {
  WString h = "Hello world";
  std::wstring sh = h.ToStd();
  DDUMP(sh);
  h = sh;
  DDUMP(h);
  Value v = sh;
  DDUMP(v);
  sh = v.ToWStd();
  DDUMP(sh);
 }
 CheckLogEtalon();
}
```