SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [File List](http://docs.google.com/files.htm)
* [include](http://docs.google.com/dir_f3190241575fd2bd132a392ae6942f4a.htm)
* [SFML](http://docs.google.com/dir_692f376662c82a26cfe4cfa3aceebe24.htm)
* [System](http://docs.google.com/dir_60c5c649f8df3b69a45a020d59f81335.htm)

Utf.hpp

1

2 //

3 // SFML - Simple and Fast Multimedia Library

4 // Copyright (C) 2007-2013 Laurent Gomila (laurent.gom@gmail.com)

5 //

6 // This software is provided 'as-is', without any express or implied warranty.

7 // In no event will the authors be held liable for any damages arising from the use of this software.

8 //

9 // Permission is granted to anyone to use this software for any purpose,

10 // including commercial applications, and to alter it and redistribute it freely,

11 // subject to the following restrictions:

12 //

13 // 1. The origin of this software must not be misrepresented;

14 // you must not claim that you wrote the original software.

15 // If you use this software in a product, an acknowledgment

16 // in the product documentation would be appreciated but is not required.

17 //

18 // 2. Altered source versions must be plainly marked as such,

19 // and must not be misrepresented as being the original software.

20 //

21 // 3. This notice may not be removed or altered from any source distribution.

22 //

24

25 #ifndef SFML\_UTF\_HPP

26 #define SFML\_UTF\_HPP

27

29 // Headers

31 #include <SFML/Config.hpp>

32 #include <algorithm>

33 #include <locale>

34 #include <string>

35 #include <cstdlib>

36

37

38 namespace sf

39 {

40 template <unsigned int N>

[41](http://docs.google.com/classUtf.htm) class [Utf](http://docs.google.com/classUtf.htm);

42

47 template <>

[48](http://docs.google.com/classsf_1_1Utf_3_018_01_4.htm) class [Utf](http://docs.google.com/classsf_1_1Utf.htm)<8>

49 {

50 public :

51

66  template <typename In>

67  static In decode(In begin, In end, Uint32& output, Uint32 replacement = 0);

68

82  template <typename Out>

83  static Out encode(Uint32 input, Out output, Uint8 replacement = 0);

84

97  template <typename In>

98  static In next(In begin, In end);

99

113  template <typename In>

114  static std::size\_t count(In begin, In end);

115

130  template <typename In, typename Out>

131  static Out fromAnsi(In begin, In end, Out output, const std::locale& locale = std::locale());

132

143  template <typename In, typename Out>

144  static Out fromWide(In begin, In end, Out output);

145

156  template <typename In, typename Out>

157  static Out fromLatin1(In begin, In end, Out output);

158

174  template <typename In, typename Out>

175  static Out toAnsi(In begin, In end, Out output, char replacement = 0, const std::locale& locale = std::locale());

176

188  template <typename In, typename Out>

189  static Out toWide(In begin, In end, Out output, wchar\_t replacement = 0);

190

202  template <typename In, typename Out>

203  static Out toLatin1(In begin, In end, Out output, char replacement = 0);

204

220  template <typename In, typename Out>

221  static Out toUtf8(In begin, In end, Out output);

222

233  template <typename In, typename Out>

234  static Out toUtf16(In begin, In end, Out output);

235

246  template <typename In, typename Out>

247  static Out toUtf32(In begin, In end, Out output);

248 };

249

254 template <>

[255](http://docs.google.com/classsf_1_1Utf_3_0116_01_4.htm) class [Utf](http://docs.google.com/classsf_1_1Utf.htm)<16>

256 {

257 public :

258

273  template <typename In>

274  static In decode(In begin, In end, Uint32& output, Uint32 replacement = 0);

275

289  template <typename Out>

290  static Out encode(Uint32 input, Out output, Uint16 replacement = 0);

291

304  template <typename In>

305  static In next(In begin, In end);

306

320  template <typename In>

321  static std::size\_t count(In begin, In end);

322

337  template <typename In, typename Out>

338  static Out fromAnsi(In begin, In end, Out output, const std::locale& locale = std::locale());

339

350  template <typename In, typename Out>

351  static Out fromWide(In begin, In end, Out output);

352

363  template <typename In, typename Out>

364  static Out fromLatin1(In begin, In end, Out output);

365

381  template <typename In, typename Out>

382  static Out toAnsi(In begin, In end, Out output, char replacement = 0, const std::locale& locale = std::locale());

383

395  template <typename In, typename Out>

396  static Out toWide(In begin, In end, Out output, wchar\_t replacement = 0);

397

409  template <typename In, typename Out>

410  static Out toLatin1(In begin, In end, Out output, char replacement = 0);

411

422  template <typename In, typename Out>

423  static Out toUtf8(In begin, In end, Out output);

424

440  template <typename In, typename Out>

441  static Out toUtf16(In begin, In end, Out output);

442

453  template <typename In, typename Out>

454  static Out toUtf32(In begin, In end, Out output);

455 };

456

461 template <>

[462](http://docs.google.com/classsf_1_1Utf_3_0132_01_4.htm) class [Utf](http://docs.google.com/classsf_1_1Utf.htm)<32>

463 {

464 public :

465

481  template <typename In>

482  static In decode(In begin, In end, Uint32& output, Uint32 replacement = 0);

483

498  template <typename Out>

499  static Out encode(Uint32 input, Out output, Uint32 replacement = 0);

500

513  template <typename In>

514  static In next(In begin, In end);

515

528  template <typename In>

529  static std::size\_t count(In begin, In end);

530

545  template <typename In, typename Out>

546  static Out fromAnsi(In begin, In end, Out output, const std::locale& locale = std::locale());

547

558  template <typename In, typename Out>

559  static Out fromWide(In begin, In end, Out output);

560

571  template <typename In, typename Out>

572  static Out fromLatin1(In begin, In end, Out output);

573

589  template <typename In, typename Out>

590  static Out toAnsi(In begin, In end, Out output, char replacement = 0, const std::locale& locale = std::locale());

591

603  template <typename In, typename Out>

604  static Out toWide(In begin, In end, Out output, wchar\_t replacement = 0);

605

617  template <typename In, typename Out>

618  static Out toLatin1(In begin, In end, Out output, char replacement = 0);

619

630  template <typename In, typename Out>

631  static Out toUtf8(In begin, In end, Out output);

632

643  template <typename In, typename Out>

644  static Out toUtf16(In begin, In end, Out output);

645

661  template <typename In, typename Out>

662  static Out toUtf32(In begin, In end, Out output);

663

677  template <typename In>

678  static Uint32 decodeAnsi(In input, const std::locale& locale = std::locale());

679

692  template <typename In>

693  static Uint32 decodeWide(In input);

694

710  template <typename Out>

711  static Out encodeAnsi(Uint32 codepoint, Out output, char replacement = 0, const std::locale& locale = std::locale());

712

727  template <typename Out>

728  static Out encodeWide(Uint32 codepoint, Out output, wchar\_t replacement = 0);

729 };

730

731 #include <SFML/System/Utf.inl>

732

733 // Make typedefs to get rid of the template syntax

734 typedef [Utf<8>](http://docs.google.com/classsf_1_1Utf_3_018_01_4.htm) [Utf8](http://docs.google.com/classsf_1_1Utf_3_018_01_4.htm);

735 typedef [Utf<16>](http://docs.google.com/classsf_1_1Utf_3_0116_01_4.htm) [Utf16](http://docs.google.com/classsf_1_1Utf_3_0116_01_4.htm);

736 typedef [Utf<32>](http://docs.google.com/classsf_1_1Utf_3_0132_01_4.htm) [Utf32](http://docs.google.com/classsf_1_1Utf_3_0132_01_4.htm);

737

738 } // namespace sf

739

740

741 #endif // SFML\_UTF\_HPP

742

743

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::