NourUtilString

1.0.0

Generated by Doxygen 1.9.5

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 MyString Namespace Reference

Classes

- class It
- class String

Functions

- String operator+ (const String &string1, const String &string2)
- bool operator== (const String &string1, const String &string2)
- std::ostream & operator<< (std::ostream &iostream, const MyString::String &string)

4.1.1 Function Documentation

4.1.1.1 operator+()

```
String MyString::operator+ (
            const String & string1,
            const String & string2 )
```

Free operator methods

4.1.1.2 operator<<()

```
std::ostream & MyString::operator<< (
            std::ostream & iostream,
            const MyString::String & string )
```

Friendship to implement the ostream operator <<

4.1.1.3 operator==()

```
bool MyString::operator== (
             const String & string1,
             const String & string2 )
```

4.2 util Namespace Reference

Classes

- class string

Functions

- std::ostream & operator<< (std::ostream &iostream, const util::string &myString)
- bool operator== (const std::string &lhsString, const util::string &rhsString)
- bool operator== (const char ∗lhsCharArray, const util::string &rhsString)
- bool operator!= (const std::string &lhsString, const util::string &rhsString)
- bool operator!= (const char ∗lhsCharArray, const util::string &rhsString)
- void concat (char ∗rawCharTarget, char ∗rawCharSource, size_t startPosition)
- void deepCopy (char ∗rawCharTarget, const char ∗rawCharSource, size_t destStartPosition, size_t srcEnd↩ Position)
- void printHeader (const char ∗text)
- void printSubHeader (const char ∗text)

4.2.1 Function Documentation

4.2.1.1 concat()

```
void util::concat (
             char * rawCharTarget,
             char * rawCharSource,
             size_t startPosition )
```

fill rawCharTarget with rawCharSource starting from startPosition

Notes:

> rawCharTarget contents will be changed the rawCharTarget is assumed to be big enough to hold the rawCharSource (i.e., its size is larger than or equal to that of the rawCharSource)

Definition at line 357 of file utilstring.cpp.

4.2.1.2 deepCopy()

```
void util::deepCopy (
            char * rawCharTarget,
            const char * rawCharSource,
            size_t destStartPosition,
            size_t srcEndPosition )
```

Definition at line 378 of file utilstring.cpp.

4.2.1.3 operator"!=() [1/2]

```
bool util::operator!= (
            const char * lhsCharArray,
            const util::string & rhsString )
```

Definition at line 342 of file utilstring.cpp.

4.2.1.4 operator"!=() [2/2]

```
bool util::operator!= (
            const std::string & lhsString,
            const util::string & rhsString )
```

Definition at line 337 of file utilstring.cpp.

4.2.1.5 operator<<()

```
std::ostream & util::operator<< (
            std::ostream & iostream,
            const util::string & myString )
```

Definition at line 319 of file utilstring.cpp.

4.2.1.6 operator==() [1/2]

```
bool util::operator== (
            const char * lhsCharArray,
            const util::string & rhsString )
```

Definition at line 331 of file utilstring.cpp.

4.2.1.7 operator==() [2/2]

```
bool util::operator== (
            const std::string & lhsString,
            const util::string & rhsString )
```

Definition at line 326 of file utilstring.cpp.

4.2.1.8 printHeader()

```
void util::printHeader (
            const char * text )
```

Definition at line 394 of file utilstring.cpp.

4.2.1.9 printSubHeader()

```
void util::printSubHeader (
            const char * text )
```

Definition at line 408 of file utilstring.cpp.

Chapter 5

Class Documentation

5.1 MyString::It< T > Class Template Reference

```
#include <String.h>
```

Public Member Functions

- It (T ∗data)
- T & operator∗ ()
- It< T > & operator++ ()
- bool operator== (const It< T > &a)
- bool operator!= (const It< T > &a)

5.1.1 Detailed Description

template<typename T>
class MyString::It< T >

Definition at line 9 of file String.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 It()

```
template<typename T >
MyString::It< T >::It (
            T * data )  [inline]
```

Definition at line 13 of file String.h.

5.1.3 Member Function Documentation

5.1.3.1 operator"!=()

```
template<typename T >
bool MyString::It< T >::operator!= (
             const It< T > & a )  [inline]
```

Definition at line 30 of file String.h.

5.1.3.2 operator*()

```
template<typename T >
T & MyString::It< T >::operator* ( )  [inline]
```

Definition at line 17 of file String.h.

5.1.3.3 operator++()

```
template<typename T >
It< T > & MyString::It< T >::operator++ ( )  [inline]
```

Definition at line 21 of file String.h.

5.1.3.4 operator==()

```
template<typename T >
bool MyString::It< T >::operator== (
             const It< T > & a )  [inline]
```

Definition at line 26 of file String.h.

The documentation for this class was generated from the following file:

- String.h

5.2 MyString::String Class Reference

```
#include <String.h>
```

Public Member Functions

- String (const char ∗data)
- String (size_t init_size=20)
- String (const String &string)
- ∼String ()
- char ∗ getText ()
- void setText (const char ∗text)
- size_t getLength () const
- void setLength (size_t size)
- void add (const String &text)
- char get (size_t pos) const
- bool compare (const String &string) const
- char operator[ ] (size_t pos)
- It< char > begin ()
- It< char > end ()

Friends

- std::ostream & operator<< (std::ostream &iostream, const MyString::String &string)

5.2.1 Detailed Description

Definition at line 36 of file String.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 String() [1/3]

```
MyString::String::String (
             const char * data )
```

5.2.2.2 String() [2/3]

```
MyString::String::String (
             size_t init_size = 20 )
```

5.2.2.3 String() [3/3]

```
MyString::String::String (
             const String & string )
```

5.2.2.4 ∼String()

```
MyString::String::∼String ( )
```

5.2.3 Member Function Documentation

5.2.3.1 add()

```
void MyString::String::add (
             const String & text )
```

5.2.3.2 begin()

```
It< char > MyString::String::begin ( )  [inline]
```

Iterator

Definition at line 82 of file String.h.

5.2.3.3 compare()

```
bool MyString::String::compare (
             const String & string ) const
```

5.2.3.4 end()

```
It< char > MyString::String::end ( )  [inline]
```

Definition at line 87 of file String.h.

5.2.3.5 get()

```
char MyString::String::get (
             size_t pos ) const
```

5.2.3.6 getLength()

```
size_t MyString::String::getLength ( ) const
```

Returns the current string length (\0 is the end of the string)

5.2.3.7 getText()

```
char * MyString::String::getText ( )
```

5.2.3.8 operator[]()

```
char MyString::String::operator[ ] (
             size_t pos )
```

Operators

5.2.3.9 setLength()

```
void MyString::String::setLength (
             size_t size )
```

Set the length of the string If the new length is smaller than its data, everything beyond will be cur If the new length is bigger, all elements will be initialized with \0

5.2.3.10 setText()

```
void MyString::String::setText (
             const char * text )
```

5.2.4 Friends And Related Function Documentation

5.2.4.1 operator<<

```
std::ostream & operator<< (
             std::ostream & iostream,
             const MyString::String & string )  [friend]
```

Friendship to implement the ostream operator <<

The documentation for this class was generated from the following file:

- String.h

5.3 util::string Class Reference

```
#include <utilstring.h>
```

Public Member Functions

- string (size_t intialSize=INITIAL_SIZE)
- string (const char ∗)
- string (const std::string &)
- string (const string &)
- ∼string (void)
- void intialize_string (size_t length=0)
- void deepCopy (const char ∗rawChar, size_t startPosition=0)
- string substr (size_t pos, size_t length)
- char ∗ c_str () const
- bool compare (const char ∗charArray) const
- void clear ()
- size_t size () const
- size_t length () const
- string & operator+ (const string &rhsString)
- string & operator+ (const char ∗strInstance)
- string & operator= (const string &rhsString)
- string & operator= (const char ∗rhsCharArray)
- string & operator= (const std::string &rhsString)
- bool operator== (const string &rhsString)
- bool operator== (const std::string &rhsString)
- bool operator== (const char ∗charArray)
- bool operator!= (const string &rhsString)
- bool operator!= (const std::string &rhsString)
- bool operator!= (const char ∗charArray)
- const char operator[ ] (size_t position)

Static Public Member Functions

- static int compare (const char ∗s1, const char ∗s2)
- static size_t rawSize (const char ∗rawChar)

Friends

- std::ostream & operator<< (std::ostream &iostream, const util::string &myString)
- bool operator== (const std::string &lhsString, const util::string &rhsString)
- bool operator== (const char ∗lhsCharArray, const util::string &rhsString)
- bool operator!= (const std::string &lhsString, const util::string &rhsString)
- bool operator!= (const char ∗lhsCharArray, const util::string &rhsString)

5.3.1 Detailed Description

Definition at line 35 of file utilstring.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 string() [1/4]

```
util::string::string (
             size_t intialSize = INITIAL_SIZE )
```

default constructor with empty initialization

Default Constructor

Definition at line 26 of file utilstring.cpp.

5.3.2.2 string() [2/4]

```
util::string::string (
             const char * data )
```

Constructor with parameter const char*

Constructor with char*

Definition at line 33 of file utilstring.cpp.

5.3.2.3 string() [3/4]

```
util::string::string (
             const std::string & data )
```

Constructor with parameter std::string

Definition at line 40 of file utilstring.cpp.

5.3.2.4 string() [4/4]

```
util::string::string (
             const string & data )
```

Copy constructor: Creates a deep copy of a passed string

Definition at line 47 of file utilstring.cpp.

5.3.2.5 ∼string()

```
util::string::∼string (
             void )
```

Definition at line 56 of file utilstring.cpp.

5.3.3 Member Function Documentation

5.3.3.1 c_str()

```
char ∗ util::string::c_str ( ) const
```

Definition at line 125 of file utilstring.cpp.

5.3.3.2 clear()

```
void util::string::clear ( )
```

5.3.3.3 compare() [1/2]

```
bool util::string::compare (
             const char ∗ charArray ) const
```

5.3.3.4 compare() [2/2]

```
int util::string::compare (
             const char ∗ lhsCharArray,
             const char ∗ rhsCharArray ) [static]
```

Compares two char ∗ strings lexicographically This function is my own implementation of the std::strcmp() function. Note this function performs a binary comparison of the ASCII code of the characters.

Parameters

str1	primitive C string to be compared.
str2	primitive C string to be compared with.

Returns

an integral value indicating the relationship between the strings: <0 : the first character that does not match has a lower value in ptr1 than in ptr2 0 : the contents of both strings are equal >0 : the first character that does not match has a greater value in ptr1 than in ptr2

Definition at line 160 of file utilstring.cpp.

5.3.3.5 deepCopy()

```
void util::string::deepCopy (
             const char * rawChar,
             size_t startPosition = 0 )
```

Design and implementation of a concept to extend the internal memory if util::string has to store more than the default INITIAL_SIZE characters note that the function copy the passed char array starting from the startPosition (i.e. it can write starting from any position in the internal string buffer) startPosition default is 0

Definition at line 101 of file utilstring.cpp.

5.3.3.6 intialize_string()

```
void util::string::intialize_string (
             size_t length = 0 )
```

Definition at line 65 of file utilstring.cpp.

5.3.3.7 length()

```
size_t util::string::length ( ) const
```

Definition at line 78 of file utilstring.cpp.

5.3.3.8 operator"!=() [1/3]

```
bool util::string::operator!= (
             const char * charArray )
```

Definition at line 237 of file utilstring.cpp.

5.3.3.9 operator"!=() [2/3]

```
bool util::string::operator!= (
             const std::string & rhsString )
```

Definition at line 230 of file utilstring.cpp.

5.3.3.10 operator"!=() [3/3]

```
bool util::string::operator!= (
             const string & rhsString )
```

Definition at line 224 of file utilstring.cpp.

5.3.3.11 operator+() [1/2]

```
string & util::string::operator+ (
             const char * rhsString )
```

concatenating util::string and const char*

Definition at line 275 of file utilstring.cpp.

5.3.3.12 operator+() [2/2]

```
string & util::string::operator+ (
             const string & rhsString )
```

Definition at line 252 of file utilstring.cpp.

5.3.3.13 operator=() [1/3]

```
string & util::string::operator= (
             const char * rhsCharArray )
```

Definition at line 191 of file utilstring.cpp.

5.3.3.14 operator=() [2/3]

```
string & util::string::operator= (
             const std::string & rhsString )
```

Definition at line 196 of file utilstring.cpp.

5.3.3.15 operator=() [3/3]

```
string & util::string::operator= (
             const string & rhsString )
```

Definition at line 186 of file utilstring.cpp.

5.3.3.16 operator==() [1/3]

```
bool util::string::operator== (
             const char * charArray )
```

Definition at line 216 of file utilstring.cpp.

5.3.3.17 operator==() [2/3]

```
bool util::string::operator== (
             const std::string & rhsString )
```

Definition at line 209 of file utilstring.cpp.

5.3.3.18 operator==() [3/3]

```
bool util::string::operator== (
             const string & rhsString )
```

Definition at line 203 of file utilstring.cpp.

5.3.3.19 operator[]()

```
const char util::string::operator[] (
             size_t position )
```

Definition at line 245 of file utilstring.cpp.

5.3.3.20 rawSize()

```
size_t util::string::rawSize (
             const char * rawChar )  [static]
```

Definition at line 83 of file utilstring.cpp.

5.3.3.21 size()

```
size_t util::string::size ( ) const
```

Get the length of the string

Definition at line 74 of file utilstring.cpp.

5.3.3.22 substr()

```
string util::string::substr (
             size_t pos,
             size_t length )
```

Definition at line 137 of file utilstring.cpp.

5.3.4 Friends And Related Function Documentation

5.3.4.1 operator"!= [1/2]

```
bool operator!= (
             const char * lhsCharArray,
             const util::string & rhsString )  [friend]
```

Definition at line 342 of file utilstring.cpp.

5.3.4.2 operator"!= [2/2]

```
bool operator!= (
             const std::string & lhsString,
             const util::string & rhsString )  [friend]
```

Definition at line 337 of file utilstring.cpp.

5.3.4.3 operator<<

```
std::ostream & operator<< (
             std::ostream & iostream,
             const util::string & myString )   [friend]
```

Definition at line 319 of file utilstring.cpp.

5.3.4.4 operator== [1/2]

```
bool operator== (
             const char * lhsCharArray,
             const util::string & rhsString )   [friend]
```

Definition at line 331 of file utilstring.cpp.

5.3.4.5 operator== [2/2]

```
bool operator== (
             const std::string & lhsString,
             const util::string & rhsString )   [friend]
```

Definition at line 326 of file utilstring.cpp.

The documentation for this class was generated from the following files:

- utilstring.h
- utilstring.cpp

Chapter 6

File Documentation

6.1 main.cpp File Reference

```
#include "utilstring.h"
#include <iostream>
```

Functions

- int main ()

6.1.1 Function Documentation

6.1.1.1 main()

```
int main ( )
```

Definition at line 88 of file main.cpp.

6.2 main.cpp

Go to the documentation of this file.

```
// Nour Ahmed
// Matrikal-Nr.: 5200991
// Assignment 1 - Own string Class

//#include "String.h"
#include "utilstring.h"

#include <iostream>



using namespace std;

```

```

//
//
//util::string concat(const char* lhsString, const char* rhsString) {
//
//
//    // a temporary object to fill it with the concatenated strings
//    // total size of the two strings combined
//    size_t lhsSize = util::string::rawSize(lhsString);
//    size_t rhsSize = util::string::rawSize(rhsString);
//
//    char* tempData = new char[lhsSize + rhsSize + 1];
//
//    // copy the passed array to the newly allocated tempData
//    // copy the lhs string from the beginning of the string then the rhs string there after
//    // ------------------------------------------------------------
//    // |0| ... |lhs size| ...  |lhs size|rhs size + lhs size + 1|
//    // |   lhs string   |   rhs string  | \0 |
//    // ------------------------------------------------------------
//    // note that at the given position the lhs \0 termination will be overwritten as this copy starts at its position
//
//    int j = 0;
//    while (lhsString[j] != '\0') {
//        tempData[j] = lhsString[j];
//        j++;
//    }
//
//    int i = 0;
//    while (rhsString[i] != '\0') {
//        tempData[j] = rhsString[i];
//        j++;
//        i++;
//    }
//    tempData[j] = '\0'; // ensure destination string is null terminated
//
//    std::cout « " > [tempData] : " « tempData « std::endl;
//
//    util::string string3(tempData);
//
//    return string3;
//
//}
//
//
//














int main() {
    char charArray[] = "text in a const char array";
    std::string stdString("another text in a std::string");

    // instantiate objects
    util::string string1;
    util::string string2("initializing with const char array");
    util::string string3(charArray);
    util::string string4(stdString);
    util::string string5(string4);

    // header ----------------------------------------------------------------
  /*  for (int k = 1; k < 255; k++)
    {
        std::cout « "\n";
        std::cout « k « "\x1B[" « k « "m  Texting\033[0m\t\t";
    }*/


    util::printHeader("NourUtilString Application");
    cout « "- Nour Ahmed                                                 -" « endl;
    cout « "- Matrikal-Nr.: 5200991                                      -" « endl;
    cout « "- Assignment 1 - Own string Class                            -" « endl;
    std::cout « "--------------------------------------------------------------------------------\n\n";

    // Test Object Instantiation ---------------------------------------------
    util::printSubHeader("Variable used for testing and their values");
    cout « "Variable used for testing and their values" « endl;

    cout « " > [charArray]  charArray (size: " « util::string::rawSize(charArray) « ")   : " « charArray « endl;
    cout « " > [stdString]  charArray (size: " « stdString.length() « ")   : " « stdString « endl;

    std::cout « "-------------------------------------------------------------------------------\n\n";

    util::printSubHeader("Test object constructors and initialization");
    cout « " > [test with default constructors]  string1 (size: " « string1.size() « ")   : " « string1 « endl;
    cout « " > [test with const char*]           string2 (size: " « string2.size() « ") : " « string2 « endl;
    cout « " > [test with std::string]           string3 (size: " « string3.size() « ") : " « string3 « endl;
    cout « " > [test with char array]            string4 (size: " « string4.size() « ") : " « string4 « endl;
    cout « " > [test with util::string]          string5 (size: " « string5.size() « ") : " « string4 « endl;
    std::cout « "-------------------------------------------------------------------------------\n\n";


    // Test member methods ---------------------------------------------------------
    util::printSubHeader("Test Member Methods");
    util::string temp = string2.substr(3, 5);
    cout « " > [test substr()]           string2 (size: " « string2.size() « ") : " « string2 « endl;
    cout « " > [test substr()]           substr(3,5) (size: " « temp.size() « ") : " « temp « endl;



    std::cout « "-------------------------------------------------------------------------------\n\n";

    // Test operators ---------------------------------------------------------
    util::printSubHeader("Test operators");

    string1 = string2;
    string2 = "more text for testing";
    string3 = std::string("text for std::string assignment");

    cout « "[test = operator util::string = util::string]\n\t string1 = string2 -> string1 (size: " « string1.size() « "): " « string1 « endl;
    cout « "[test = operator util::string = const char*]\n\t string2 = \"...\" -> string2 (size: " « string2.size() « "): " « string2 « endl;
    cout « "[test = operator util::string = std::string]\n\t string3 = std::string(\"...\") -> string3 (size: " « string3.size() « "): " « string3 « endl;

    std::cout « "-------------------------------------------------------------------------------\n\n";

    string1 = string2;
    cout « "[test == operator util::string == util::string]\n\t string1 == string2   -> " « ((string1 == string2)?"true":"false") « endl;

    cout « "[test == operator util::string == std::string]\n\t string1 == stdString -> " « ((string1 == stdString) ? "true" : "false") « endl;
    cout « "[test == operator std::string == util::string]\n\t stdString == string1 -> " « ((stdString == string1) ? "true" : "false") « endl;

    cout « "[test == operator util::string == const char*]\n\t string1 == charArray -> " « ((string1 == charArray) ? "true" : "false") « endl;
    cout « "[test == operator const char* == util::string]\n\t charArray == string1 -> " « ((charArray == string1) ? "true" : "false") « endl;

    std::cout « "-------------------------------------------------------------------------------\n\n";

    cout « "[test != operator util::string != util::string]\n\t string1 != string2   -> " « ((string1 != string2) ? "true" : "false") « endl;

    cout « "[test != operator util::string != std::string]\n\t string1 != stdString -> " « ((string1 != stdString) ? "true" : "false") « endl;
    cout « "[test != operator std::string != util::string]\n\t stdString != string1 -> " « ((stdString != string1) ? "true" : "false") « endl;

    cout « "[test != operator util::string != const char*]\n\t string1 != charArray -> " « ((string1 != charArray) ? "true" : "false") « endl;
    cout « "[test != operator const char* != util::string]\n\t charArray != string1 -> " « ((charArray != string1) ? "true" : "false") « endl;
    std::cout « "-------------------------------------------------------------------------------\n\n";

    cout « "[test [] operator util::string[]]" « endl;
    cout « "\tstring1: " « string1 « "-> string1[0]: " « string1[0]« endl;
    cout « "\tstring2: " « string2 « "-> string2[3]: " « string2[3] « endl;
    cout « "\tstring3: " « string3 « "-> string3[50]: " « string3[50] « endl;
    std::cout « "-------------------------------------------------------------------------------\n\n";

    //cout « " > [util::string + const char*]  string4 + charArray: " « string4 + charArray « endl;
    string2= string2 + string3;
    cout « "string2 (size: " « string2.size() « ") : " « string2 « endl;
    cout « "string3 (size: " « string3.size() « ") : " « string3 « endl;
    cout « " > [util::string + util::string]  string2 + string3: " « string2 + string3
         « "\nstring2 (size: " « string2.size() « ") : " « string2 « endl;


    std::cout « "-------------------------------------------------------------------------------\n\n";

    cout « "[test « operator std::cout « util::string « int « std::string « char *]:\n\t"
        « string1 « ", "
        « "size: "
        « string1.size() « ", "
        « stdString « ", "
        « charArray
        « endl;



    //cout « " > [util::string + const char*]  string4 + charArray: " « string4 + charArray « endl;

    //util::string string6(string4 + charArray);

    //cout « " > [test with util::string]          string6 (size: " « string6.size() « ") : " «
 string6 « endl;

    std::cout « "-------------------------------------------------------------------------------\n\n";

   //cout « " > [util::string + const char*]: " « util::concat(string4.c_str(), charArray) « endl;

    std::cout « "-------------------------------------------------------------------------------\n\n";

    char s1[100] = "programming ", s2[] = "is awesome";
    cout « "s1: " « s1 « endl;
    cout « "s2: " « s2 « endl;

    /*util::concat(s1, s2, 2);
    cout « "s1: " « s1 « endl;

    util::concat(s1, s2);
    cout « "s1: " « s1 « endl;

    util::deepCopy(s1, s2, 3 , 10);
    cout « "s1: " « s1 « endl;*/

    util::deepCopy(s1, s2);
    cout « "s1: " « s1 « endl;


 /*
    delete[] a;
    delete[] b;*/

    //MyString::String string1;
    //MyString::String string2("Welt");


    //string1.setText("Hallo ");


    //cout « "String1 and String2: " « string1 « string2 « endl;
    //cout « "String1 + String2: " « pstring « endl;
    //cout « "Comparing 1 with 2: " « (string1 == string2) « endl;
    //cout « "Comparing 1 with 1: " « (string1 == string1) « endl;
    //cout « "Get: " « string1[0] « endl;

    //MyString::It<char> it = pstring.begin();
    //while (it != pstring.end()) {
    //    cout « *it « endl;
    //    ++it;
    //}


    return 0;
}
```

6.3 main.h File Reference

```
#include <iostream>
```

6.3.1 Detailed Description

==

Author

: Nour Ahmed @email : nahmed@stud.hs-bremen.de, nour @repo : @createdOn : 23.11.2022 @description : Defines the entry point for the application

In this application the class util::string is used and tested. Each method and operator is tested with all possible uasges (e.g., concatenating different strings etc.) Note: For this task no error handling is required.Example↩ : Accessing

6.3.2 an invalid index by using operator [ ]

Definition in file main.h.

6.4 main.h

Go to the documentation of this file.

```
// CMakeProject1.h : Include file for standard system include files,
// or project specific include files.

#pragma once

#include <iostream>

// TODO: Reference additional headers your program requires here.

```

6.5 out/build/x64-Debug/CMakeFiles/3.20.21032501-MSVC_2/Compiler↩ IdC/CMakeCCompilerId.c File Reference

Macros

- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_DIALECT

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_dialect_default

6.5.1 Macro Definition Documentation

6.5.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 613 of file CMakeCCompilerId.c.

6.5.1.2 C_DIALECT

```
#define C_DIALECT
```

Definition at line 697 of file CMakeCCompilerId.c.

6.5.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 360 of file CMakeCCompilerId.c.

6.5.1.4 DEC

```
#define DEC(
             n )
```

Value:

```
('0' + (((n) / 10000000)%10)), \
('0' + (((n) / 1000000)%10)),  \
('0' + (((n) / 100000)%10)),   \
('0' + (((n) / 10000)%10)),    \
('0' + (((n) / 1000)%10)),     \
('0' + (((n) / 100)%10)),      \
('0' + (((n) / 10)%10)),       \
('0' +  ((n) % 10))
```

Definition at line 617 of file CMakeCCompilerId.c.

6.5.1.5 HEX

```
#define HEX(
              n )
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 628 of file CMakeCCompilerId.c.

6.5.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 485 of file CMakeCCompilerId.c.

6.5.1.7 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 381 of file CMakeCCompilerId.c.

6.5.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 380 of file CMakeCCompilerId.c.

6.5.2 Function Documentation

6.5.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 717 of file CMakeCCompilerId.c.

6.5.3 Variable Documentation

6.5.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 688 of file CMakeCCompilerId.c.

6.5.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 367 of file CMakeCCompilerId.c.

6.5.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
=
  "INFO" ":" "dialect_default[" C_DIALECT "]"
```

Definition at line 706 of file CMakeCCompilerId.c.

6.5.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 687 of file CMakeCCompilerId.c.

6.6 CMakeCCompilerId.c

Go to the documentation of this file.

```
#ifdef __cplusplus
# error "A C++ compiler has been selected for C."
#endif

#if defined(__18CXX)
# define ID_VOID_MAIN
#endif
#if defined(__CLASSIC_C__)
/* cv-qualifiers did not exist in K&R C */
# define const
# define volatile
#endif


/* Version number components: V=Version, R=Revision, P=Patch
   Version date components:   YYYY=Year, MM=Month,   DD=Day  */

#if defined(__INTEL_COMPILER) || defined(__ICC)
# define COMPILER_ID "Intel"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# if defined(__GNUC__)
#  define SIMULATE_ID "GNU"
# endif
  /* __INTEL_COMPILER = VRP */
# define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
# if defined(__INTEL_COMPILER_UPDATE)
#  define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
# else
#  define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
# endif
# if defined(__INTEL_COMPILER_BUILD_DATE)
  /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
#  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
# endif
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# if defined(__GNUC__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
# elif defined(__GNUG__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
# define COMPILER_ID "IntelLLVM"
#if defined(_MSC_VER)
# define SIMULATE_ID "MSVC"
#endif
#if defined(__GNUC__)
# define SIMULATE_ID "GNU"
#endif
/* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
 * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
 * VVVV is no smaller than the current year when a versio is released.
 */
#if __INTEL_LLVM_COMPILER < 1000000L
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
#else
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
#endif
#if defined(_MSC_VER)
  /* _MSC_VER = VVRR */
# define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
# define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
#endif
#if defined(__GNUC__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
```

```
#elif defined(__GNUG__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
#endif
#if defined(__GNUC_MINOR__)
# define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#endif
#if defined(__GNUC_PATCHLEVEL__)
# define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#endif

#elif defined(__PATHCC__)
# define COMPILER_ID "PathScale"
# define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
# define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
# if defined(__PATHCC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
# endif

#elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
# define COMPILER_ID "Embarcadero"
# define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
# define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
# define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)

#elif defined(__BORLANDC__)
# define COMPILER_ID "Borland"
  /* __BORLANDC__ = 0xVRR */
# define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
# define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)

#elif defined(__WATCOMC__) && __WATCOMC__ < 1200
# define COMPILER_ID "Watcom"
   /* __WATCOMC__ = VVRR */
# define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__WATCOMC__)
# define COMPILER_ID "OpenWatcom"
   /* __WATCOMC__ = VVRP + 1100 */
# define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__SUNPRO_C)
# define COMPILER_ID "SunPro"
# if __SUNPRO_C >= 0x5100
   /* __SUNPRO_C = 0xVRRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»12)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xFF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
# else
   /* __SUNPRO_CC = 0xVRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»8)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
# endif

#elif defined(__HP_cc)
# define COMPILER_ID "HP"
  /* __HP_cc = VVRRPP */
# define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
# define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__HP_cc     % 100)

#elif defined(__DECC)
# define COMPILER_ID "Compaq"
  /* __DECC_VER = VVRRTPPPP */
# define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
# define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000  % 100)
# define COMPILER_VERSION_PATCH DEC(__DECC_VER         % 10000)

#elif defined(__IBMC__) && defined(__COMPILER_VER__)
# define COMPILER_ID "zOS"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__ibmxl__) && defined(__clang__)
# define COMPILER_ID "XLClang"
# define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
# define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
# define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)


#elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
# define COMPILER_ID "XL"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
# define COMPILER_ID "VisualAge"
  /* __IBMC__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)

#elif defined(__NVCOMPILER)
# define COMPILER_ID "NVHPC"
# define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
# if defined(__NVCOMPILER_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
# endif

#elif defined(__PGI)
# define COMPILER_ID "PGI"
# define COMPILER_VERSION_MAJOR DEC(__PGIC__)
# define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
# if defined(__PGIC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
# endif

#elif defined(_CRAYC)
# define COMPILER_ID "Cray"
# define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
# define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)

#elif defined(__TI_COMPILER_VERSION__)
# define COMPILER_ID "TI"
  /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
# define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
# define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
# define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)

#elif defined(__FUJITSU) || defined(__FCC_VERSION) || defined(__fcc_version)
# define COMPILER_ID "Fujitsu"

#elif defined(__ghs__)
# define COMPILER_ID "GHS"
/* __GHS_VERSION_NUMBER = VVVVRP */
# ifdef __GHS_VERSION_NUMBER
# define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
# define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
# define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
# endif

#elif defined(__TINYC__)
# define COMPILER_ID "TinyCC"

#elif defined(__BCC__)
# define COMPILER_ID "Bruce"

#elif defined(__SCO_VERSION__)
# define COMPILER_ID "SCO"

#elif defined(__ARMCC_VERSION) && !defined(__clang__)
# define COMPILER_ID "ARMCC"
#if __ARMCC_VERSION >= 1000000
  /* __ARMCC_VERSION = VRRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
#else
  /* __ARMCC_VERSION = VRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
#endif


#elif defined(__clang__) && defined(__apple_build_version__)
# define COMPILER_ID "AppleClang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

#elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
# define COMPILER_ID "ARMClang"
  # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
# define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)

#elif defined(__clang__)
# define COMPILER_ID "Clang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif

#elif defined(__GNUC__)
# define COMPILER_ID "GNU"
# define COMPILER_VERSION_MAJOR DEC(__GNUC__)
# if defined(__GNUC_MINOR__)
#  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif defined(_MSC_VER)
# define COMPILER_ID "MSVC"
  /* _MSC_VER = VVRR */
# define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
# define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
# if defined(_MSC_FULL_VER)
#  if _MSC_VER >= 1400
    /* _MSC_FULL_VER = VVRRPPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
#  else
    /* _MSC_FULL_VER = VVRRPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
#  endif
# endif
# if defined(_MSC_BUILD)
#  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
# endif

#elif defined(__VISUALDSPVERSION__) || defined(__ADSPBLACKFIN__) || defined(__ADSPTS__) ||
 defined(__ADSP21000__)
# define COMPILER_ID "ADSP"
#if defined(__VISUALDSPVERSION__)
  /* __VISUALDSPVERSION__ = 0xVVRRPP00 */
# define COMPILER_VERSION_MAJOR HEX(__VISUALDSPVERSION__»24)
# define COMPILER_VERSION_MINOR HEX(__VISUALDSPVERSION__»16 & 0xFF)
# define COMPILER_VERSION_PATCH HEX(__VISUALDSPVERSION__»8  & 0xFF)
#endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# define COMPILER_ID "IAR"
# if defined(__VER__) && defined(__ICCARM__)
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
#  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
#  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
 defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
 defined(__ICC8051__) || defined(__ICCSTM8__))
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
#  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
#  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# endif

#elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
# define COMPILER_ID "SDCC"
# if defined(__SDCC_VERSION_MAJOR)
#  define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
#  define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
#  define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
# else
  /* SDCC = VRP */
#  define COMPILER_VERSION_MAJOR DEC(SDCC/100)
#  define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
#  define COMPILER_VERSION_PATCH DEC(SDCC    % 10)
# endif


/* These compilers are either not known or too old to define an
  identification macro.  Try to identify the platform and guess that
  it is the native compiler.  */
#elif defined(__hpux) || defined(__hpua)
# define COMPILER_ID "HP"

#else /* unknown compiler */
# define COMPILER_ID ""
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
#ifdef SIMULATE_ID
char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
#endif

#ifdef __QNXNTO__
char const* qnxnto = "INFO" ":" "qnxnto[]";
#endif

#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
#endif

#define STRINGIFY_HELPER(X) #X
#define STRINGIFY(X) STRINGIFY_HELPER(X)

/* Identify known platforms by name.  */
#if defined(__linux) || defined(__linux__) || defined(linux)
# define PLATFORM_ID "Linux"

#elif defined(__CYGWIN__)
# define PLATFORM_ID "Cygwin"

#elif defined(__MINGW32__)
# define PLATFORM_ID "MinGW"

#elif defined(__APPLE__)
# define PLATFORM_ID "Darwin"

#elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
# define PLATFORM_ID "Windows"

#elif defined(__FreeBSD__) || defined(__FreeBSD)
# define PLATFORM_ID "FreeBSD"

#elif defined(__NetBSD__) || defined(__NetBSD)
# define PLATFORM_ID "NetBSD"

#elif defined(__OpenBSD__) || defined(__OPENBSD)
# define PLATFORM_ID "OpenBSD"

#elif defined(__sun) || defined(sun)
# define PLATFORM_ID "SunOS"

#elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
# define PLATFORM_ID "AIX"

#elif defined(__hpux) || defined(__hpux__)
# define PLATFORM_ID "HP-UX"

#elif defined(__HAIKU__)
# define PLATFORM_ID "Haiku"

#elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
# define PLATFORM_ID "BeOS"

#elif defined(__QNX__) || defined(__QNXNTO__)
# define PLATFORM_ID "QNX"

#elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
# define PLATFORM_ID "Tru64"
```

```

#elif defined(__riscos) || defined(__riscos__)
# define PLATFORM_ID "RISCos"

#elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
# define PLATFORM_ID "SINIX"

#elif defined(__UNIX_SV__)
# define PLATFORM_ID "UNIX_SV"

#elif defined(__bsdos__)
# define PLATFORM_ID "BSDOS"

#elif defined(_MPRAS) || defined(MPRAS)
# define PLATFORM_ID "MP-RAS"

#elif defined(__osf) || defined(__osf__)
# define PLATFORM_ID "OSF1"

#elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
# define PLATFORM_ID "SCO_SV"

#elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
# define PLATFORM_ID "ULTRIX"

#elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
# define PLATFORM_ID "Xenix"

#elif defined(__WATCOMC__)
# if defined(__LINUX__)
#  define PLATFORM_ID "Linux"

# elif defined(__DOS__)
#  define PLATFORM_ID "DOS"

# elif defined(__OS2__)
#  define PLATFORM_ID "OS2"

# elif defined(__WINDOWS__)
#  define PLATFORM_ID "Windows3x"

# elif defined(__VXWORKS__)
#  define PLATFORM_ID "VxWorks"

# else /* unknown platform */
#  define PLATFORM_ID
# endif

#elif defined(__INTEGRITY)
# if defined(INT_178B)
#  define PLATFORM_ID "Integrity178"

# else /* regular Integrity */
#  define PLATFORM_ID "Integrity"
# endif

#else /* unknown platform */
# define PLATFORM_ID

#endif

/* For windows compilers MSVC and Intel we can determine
   the architecture of the compiler being used.  This is because
   the compilers do not have flags that can change the architecture,
   but rather depend on which compiler is being used
*/
#if defined(_WIN32) && defined(_MSC_VER)
# if defined(_M_IA64)
#  define ARCHITECTURE_ID "IA64"

# elif defined(_M_ARM64EC)
#  define ARCHITECTURE_ID "ARM64EC"

# elif defined(_M_X64) || defined(_M_AMD64)
#  define ARCHITECTURE_ID "x64"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# elif defined(_M_ARM64)
#  define ARCHITECTURE_ID "ARM64"

# elif defined(_M_ARM)
#  if _M_ARM == 4
#   define ARCHITECTURE_ID "ARMV4I"
#  elif _M_ARM == 5
#   define ARCHITECTURE_ID "ARMV5I"
#  else
#   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
#  endif

# elif defined(_M_MIPS)
#  define ARCHITECTURE_ID "MIPS"

# elif defined(_M_SH)
#  define ARCHITECTURE_ID "SHx"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__WATCOMC__)
# if defined(_M_I86)
#  define ARCHITECTURE_ID "I86"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# if defined(__ICCARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__ICCRX__)
#  define ARCHITECTURE_ID "RX"

# elif defined(__ICCRH850__)
#  define ARCHITECTURE_ID "RH850"

# elif defined(__ICCRL78__)
#  define ARCHITECTURE_ID "RL78"

# elif defined(__ICCRISCV__)
#  define ARCHITECTURE_ID "RISCV"

# elif defined(__ICCAVR__)
#  define ARCHITECTURE_ID "AVR"

# elif defined(__ICC430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__ICCV850__)
#  define ARCHITECTURE_ID "V850"

# elif defined(__ICC8051__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__ICCSTM8__)
#  define ARCHITECTURE_ID "STM8"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__ghs__)
# if defined(__PPC64__)
#  define ARCHITECTURE_ID "PPC64"

# elif defined(__ppc__)
#  define ARCHITECTURE_ID "PPC"

# elif defined(__ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__x86_64__)
#  define ARCHITECTURE_ID "x64"

# elif defined(__i386__)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__TI_COMPILER_VERSION__)
# if defined(__TI_ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__MSP430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__TMS320C28XX__)
#  define ARCHITECTURE_ID "TMS320C28x"

# elif defined(__TMS320C6X__) || defined(_TMS320C6X)
#  define ARCHITECTURE_ID "TMS320C6x"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#else
#  define ARCHITECTURE_ID
#endif

/* Convert integer to decimal digit literals.  */
#define DEC(n)                   \
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))

/* Convert integer to hex digit literals.  */
#define HEX(n)             \
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))

/* Construct a string literal encoding the version number components. */
#ifdef COMPILER_VERSION_MAJOR
char const info_version[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
  COMPILER_VERSION_MAJOR,
# ifdef COMPILER_VERSION_MINOR
  '.', COMPILER_VERSION_MINOR,
#  ifdef COMPILER_VERSION_PATCH
   '.', COMPILER_VERSION_PATCH,
#   ifdef COMPILER_VERSION_TWEAK
    '.', COMPILER_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct a string literal encoding the internal version number. */
#ifdef COMPILER_VERSION_INTERNAL
char const info_version_internal[] = {
  'I', 'N', 'F', 'O', ':',
  'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
  'i','n','t','e','r','n','a','l','[',
  COMPILER_VERSION_INTERNAL,']','\0'};
#endif

/* Construct a string literal encoding the version number components. */
#ifdef SIMULATE_VERSION_MAJOR
char const info_simulate_version[] = {
  'I', 'N', 'F', 'O', ':',
  's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
  SIMULATE_VERSION_MAJOR,
# ifdef SIMULATE_VERSION_MINOR
  '.', SIMULATE_VERSION_MINOR,
#  ifdef SIMULATE_VERSION_PATCH
   '.', SIMULATE_VERSION_PATCH,
#   ifdef SIMULATE_VERSION_TWEAK
    '.', SIMULATE_VERSION_TWEAK,
#   endif
#  endif
# endif
  ']','\0'};
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";
```

```



#if !defined(__STDC__)
# if (defined(_MSC_VER) && !defined(__clang__)) \
  || (defined(__ibmxl__) || defined(__IBMC__))
#  define C_DIALECT "90"
# else
#  define C_DIALECT
# endif
#elif __STDC_VERSION__ >= 201000L
# define C_DIALECT "11"
#elif __STDC_VERSION__ >= 199901L
# define C_DIALECT "99"
#else
# define C_DIALECT "90"
#endif
const char* info_language_dialect_default =
  "INFO" ":" "dialect_default[" C_DIALECT "]";

/*--------------------------------------------------------------------------*/

#ifdef ID_VOID_MAIN
void main() {}
#else
# if defined(__CLASSIC_C__)
int main(argc, argv) int argc; char *argv[];
# else
int main(int argc, char* argv[])
# endif
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
  require += info_arch[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
#endif
#ifdef COMPILER_VERSION_INTERNAL
  require += info_version_internal[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
  require += info_cray[argc];
#endif
  require += info_language_dialect_default[argc];
  (void)argv;
  return require;
}
#endif
```

6.7 out/build/x64-Debug/CMakeFiles/3.20.21032501-MSVC_2/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD __cplusplus

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_dialect_default

6.7.1 Macro Definition Documentation

6.7.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 598 of file CMakeCXXCompilerId.cpp.

6.7.1.2 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 345 of file CMakeCXXCompilerId.cpp.

6.7.1.3 CXX_STD

```
#define CXX_STD __cplusplus
```

Definition at line 690 of file CMakeCXXCompilerId.cpp.

6.7.1.4 DEC

```
#define DEC(
              n )
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 602 of file CMakeCXXCompilerId.cpp.

6.7.1.5 HEX

```
#define HEX(
              n )
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 613 of file CMakeCXXCompilerId.cpp.

6.7.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 470 of file CMakeCXXCompilerId.cpp.

6.7.1.7 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 366 of file CMakeCXXCompilerId.cpp.

6.7.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 365 of file CMakeCXXCompilerId.cpp.

6.7.2 Function Documentation

6.7.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 711 of file CMakeCXXCompilerId.cpp.

6.7.3 Variable Documentation

6.7.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 673 of file CMakeCXXCompilerId.cpp.

6.7.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 352 of file CMakeCXXCompilerId.cpp.

6.7.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
= "INFO" ":" "dialect_default["
  "98"
"]"
```

Definition at line 693 of file CMakeCXXCompilerId.cpp.

6.7.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 672 of file CMakeCXXCompilerId.cpp.

6.8 CMakeCXXCompilerId.cpp

Go to the documentation of this file.

```
/* This source file must have a .cpp extension so that all C++ compilers
   recognize the extension without flags.  Borland does not know .cxx for
   example.  */
#ifndef __cplusplus
# error "A C compiler has been selected for C++."
#endif


/* Version number components: V=Version, R=Revision, P=Patch
   Version date components:   YYYY=Year, MM=Month,   DD=Day  */

#if defined(__COMO__)
# define COMPILER_ID "Comeau"
  /* __COMO_VERSION__ = VRR */
# define COMPILER_VERSION_MAJOR DEC(__COMO_VERSION__ / 100)
# define COMPILER_VERSION_MINOR DEC(__COMO_VERSION__ % 100)

#elif defined(__INTEL_COMPILER) || defined(__ICC)
# define COMPILER_ID "Intel"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# if defined(__GNUC__)
#  define SIMULATE_ID "GNU"
# endif
  /* __INTEL_COMPILER = VRP */
# define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
# if defined(__INTEL_COMPILER_UPDATE)
#  define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
# else
#  define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
# endif
# if defined(__INTEL_COMPILER_BUILD_DATE)
  /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
#  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
# endif
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# if defined(__GNUC__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
# elif defined(__GNUG__)
#  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif (defined(__clang__) && defined(__INTEL_CLANG_COMPILER)) || defined(__INTEL_LLVM_COMPILER)
# define COMPILER_ID "IntelLLVM"
#if defined(_MSC_VER)
# define SIMULATE_ID "MSVC"
#endif
#if defined(__GNUC__)
# define SIMULATE_ID "GNU"
#endif
/* __INTEL_LLVM_COMPILER = VVVVRP prior to 2021.2.0, VVVVRRPP for 2021.2.0 and
 * later.  Look for 6 digit vs. 8 digit version number to decide encoding.
 * VVVV is no smaller than the current year when a versio is released.
 */
#if __INTEL_LLVM_COMPILER < 1000000L
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/100)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER    % 10)
#else
# define COMPILER_VERSION_MAJOR DEC(__INTEL_LLVM_COMPILER/10000)
# define COMPILER_VERSION_MINOR DEC(__INTEL_LLVM_COMPILER/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__INTEL_LLVM_COMPILER     % 100)
#endif
#if defined(_MSC_VER)
  /* _MSC_VER = VVRR */
# define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
# define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
#endif
#if defined(__GNUC__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
```

```
#elif defined(__GNUG__)
# define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
#endif
#if defined(__GNUC_MINOR__)
# define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
#endif
#if defined(__GNUC_PATCHLEVEL__)
# define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
#endif

#elif defined(__PATHCC__)
# define COMPILER_ID "PathScale"
# define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
# define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
# if defined(__PATHCC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
# endif

#elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
# define COMPILER_ID "Embarcadero"
# define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
# define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
# define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)

#elif defined(__BORLANDC__)
# define COMPILER_ID "Borland"
  /* __BORLANDC__ = 0xVRR */
# define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
# define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)

#elif defined(__WATCOMC__) && __WATCOMC__ < 1200
# define COMPILER_ID "Watcom"
   /* __WATCOMC__ = VVRR */
# define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__WATCOMC__)
# define COMPILER_ID "OpenWatcom"
   /* __WATCOMC__ = VVRP + 1100 */
# define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
# define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
# if (__WATCOMC__ % 10) > 0
#  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
# endif

#elif defined(__SUNPRO_CC)
# define COMPILER_ID "SunPro"
# if __SUNPRO_CC >= 0x5100
   /* __SUNPRO_CC = 0xVRRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC»12)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC»4 & 0xFF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
# else
   /* __SUNPRO_CC = 0xVRP */
#  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC»8)
#  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC»4 & 0xF)
#  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
# endif

#elif defined(__HP_aCC)
# define COMPILER_ID "HP"
  /* __HP_aCC = VVRRPP */
# define COMPILER_VERSION_MAJOR DEC(__HP_aCC/10000)
# define COMPILER_VERSION_MINOR DEC(__HP_aCC/100 % 100)
# define COMPILER_VERSION_PATCH DEC(__HP_aCC     % 100)

#elif defined(__DECCXX)
# define COMPILER_ID "Compaq"
  /* __DECCXX_VER = VVRRTPPPP */
# define COMPILER_VERSION_MAJOR DEC(__DECCXX_VER/10000000)
# define COMPILER_VERSION_MINOR DEC(__DECCXX_VER/100000  % 100)
# define COMPILER_VERSION_PATCH DEC(__DECCXX_VER         % 10000)

#elif defined(__IBMCPP__) && defined(__COMPILER_VER__)
# define COMPILER_ID "zOS"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__ibmxl__) && defined(__clang__)
# define COMPILER_ID "XLClang"
# define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
# define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
# define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
# define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)


#elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ >= 800
# define COMPILER_ID "XL"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ < 800
# define COMPILER_ID "VisualAge"
  /* __IBMCPP__ = VRP */
# define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
# define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
# define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)

#elif defined(__NVCOMPILER)
# define COMPILER_ID "NVHPC"
# define COMPILER_VERSION_MAJOR DEC(__NVCOMPILER_MAJOR__)
# define COMPILER_VERSION_MINOR DEC(__NVCOMPILER_MINOR__)
# if defined(__NVCOMPILER_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__NVCOMPILER_PATCHLEVEL__)
# endif

#elif defined(__PGI)
# define COMPILER_ID "PGI"
# define COMPILER_VERSION_MAJOR DEC(__PGIC__)
# define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
# if defined(__PGIC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
# endif

#elif defined(_CRAYC)
# define COMPILER_ID "Cray"
# define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
# define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)

#elif defined(__TI_COMPILER_VERSION__)
# define COMPILER_ID "TI"
  /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
# define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
# define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
# define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)

#elif defined(__FUJITSU) || defined(__FCC_VERSION) || defined(__fcc_version)
# define COMPILER_ID "Fujitsu"

#elif defined(__ghs__)
# define COMPILER_ID "GHS"
/* __GHS_VERSION_NUMBER = VVVVRP */
# ifdef __GHS_VERSION_NUMBER
# define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
# define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
# define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
# endif

#elif defined(__SCO_VERSION__)
# define COMPILER_ID "SCO"

#elif defined(__ARMCC_VERSION) && !defined(__clang__)
# define COMPILER_ID "ARMCC"
#if __ARMCC_VERSION >= 1000000
  /* __ARMCC_VERSION = VRRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
#else
  /* __ARMCC_VERSION = VRPPPP */
  # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
  # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
#endif


#elif defined(__clang__) && defined(__apple_build_version__)
# define COMPILER_ID "AppleClang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif
# define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)

#elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
# define COMPILER_ID "ARMClang"
  # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
  # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
  # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
# define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)

#elif defined(__clang__)
# define COMPILER_ID "Clang"
# if defined(_MSC_VER)
#  define SIMULATE_ID "MSVC"
# endif
# define COMPILER_VERSION_MAJOR DEC(__clang_major__)
# define COMPILER_VERSION_MINOR DEC(__clang_minor__)
# define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
# if defined(_MSC_VER)
   /* _MSC_VER = VVRR */
#  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
#  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
# endif

#elif defined(__GNUC__) || defined(__GNUG__)
# define COMPILER_ID "GNU"
# if defined(__GNUC__)
#  define COMPILER_VERSION_MAJOR DEC(__GNUC__)
# else
#  define COMPILER_VERSION_MAJOR DEC(__GNUG__)
# endif
# if defined(__GNUC_MINOR__)
#  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
# endif
# if defined(__GNUC_PATCHLEVEL__)
#  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
# endif

#elif defined(_MSC_VER)
# define COMPILER_ID "MSVC"
  /* _MSC_VER = VVRR */
# define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
# define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
# if defined(_MSC_FULL_VER)
#  if _MSC_VER >= 1400
    /* _MSC_FULL_VER = VVRRPPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
#  else
    /* _MSC_FULL_VER = VVRRPPPP */
#   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
#  endif
# endif
# if defined(_MSC_BUILD)
#  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
# endif

#elif defined(__VISUALDSPVERSION__) || defined(__ADSPBLACKFIN__) || defined(__ADSPTS__) || defined(__ADSP21000__)
# define COMPILER_ID "ADSP"
#if defined(__VISUALDSPVERSION__)
  /* __VISUALDSPVERSION__ = 0xVVRRPP00 */
# define COMPILER_VERSION_MAJOR HEX(__VISUALDSPVERSION__»24)
# define COMPILER_VERSION_MINOR HEX(__VISUALDSPVERSION__»16 & 0xFF)
# define COMPILER_VERSION_PATCH HEX(__VISUALDSPVERSION__»8  & 0xFF)
#endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# define COMPILER_ID "IAR"
# if defined(__VER__) && defined(__ICCARM__)
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
#  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
#  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) || defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) || defined(__ICC8051__) || defined(__ICCSTM8__))
#  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
#  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
#  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
#  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
# endif


/* These compilers are either not known or too old to define an
  identification macro.  Try to identify the platform and guess that
  it is the native compiler.  */
```

```
#elif defined(__hpux) || defined(__hpua)
# define COMPILER_ID "HP"

#else /* unknown compiler */
# define COMPILER_ID ""
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
#ifdef SIMULATE_ID
char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
#endif

#ifdef __QNXNTO__
char const* qnxnto = "INFO" ":" "qnxnto[]";
#endif

#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
#endif

#define STRINGIFY_HELPER(X) #X
#define STRINGIFY(X) STRINGIFY_HELPER(X)

/* Identify known platforms by name.  */
#if defined(__linux) || defined(__linux__) || defined(linux)
# define PLATFORM_ID "Linux"

#elif defined(__CYGWIN__)
# define PLATFORM_ID "Cygwin"

#elif defined(__MINGW32__)
# define PLATFORM_ID "MinGW"

#elif defined(__APPLE__)
# define PLATFORM_ID "Darwin"

#elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
# define PLATFORM_ID "Windows"

#elif defined(__FreeBSD__) || defined(__FreeBSD)
# define PLATFORM_ID "FreeBSD"

#elif defined(__NetBSD__) || defined(__NetBSD)
# define PLATFORM_ID "NetBSD"

#elif defined(__OpenBSD__) || defined(__OPENBSD)
# define PLATFORM_ID "OpenBSD"

#elif defined(__sun) || defined(sun)
# define PLATFORM_ID "SunOS"

#elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
# define PLATFORM_ID "AIX"

#elif defined(__hpux) || defined(__hpux__)
# define PLATFORM_ID "HP-UX"

#elif defined(__HAIKU__)
# define PLATFORM_ID "Haiku"

#elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
# define PLATFORM_ID "BeOS"

#elif defined(__QNX__) || defined(__QNXNTO__)
# define PLATFORM_ID "QNX"

#elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
# define PLATFORM_ID "Tru64"

#elif defined(__riscos) || defined(__riscos__)
# define PLATFORM_ID "RISCos"

#elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
# define PLATFORM_ID "SINIX"

#elif defined(__UNIX_SV__)
# define PLATFORM_ID "UNIX_SV"

#elif defined(__bsdos__)
# define PLATFORM_ID "BSDOS"

#elif defined(_MPRAS) || defined(MPRAS)
# define PLATFORM_ID "MP-RAS"
```

```

#elif defined(__osf) || defined(__osf__)
# define PLATFORM_ID "OSF1"

#elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
# define PLATFORM_ID "SCO_SV"

#elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
# define PLATFORM_ID "ULTRIX"

#elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
# define PLATFORM_ID "Xenix"

#elif defined(__WATCOMC__)
# if defined(__LINUX__)
#  define PLATFORM_ID "Linux"

# elif defined(__DOS__)
#  define PLATFORM_ID "DOS"

# elif defined(__OS2__)
#  define PLATFORM_ID "OS2"

# elif defined(__WINDOWS__)
#  define PLATFORM_ID "Windows3x"

# elif defined(__VXWORKS__)
#  define PLATFORM_ID "VxWorks"

# else /* unknown platform */
#  define PLATFORM_ID
# endif

#elif defined(__INTEGRITY)
# if defined(INT_178B)
#  define PLATFORM_ID "Integrity178"

# else /* regular Integrity */
#  define PLATFORM_ID "Integrity"
# endif

#else /* unknown platform */
# define PLATFORM_ID

#endif

/* For windows compilers MSVC and Intel we can determine
   the architecture of the compiler being used.  This is because
   the compilers do not have flags that can change the architecture,
   but rather depend on which compiler is being used
*/
#if defined(_WIN32) && defined(_MSC_VER)
# if defined(_M_IA64)
#  define ARCHITECTURE_ID "IA64"

# elif defined(_M_ARM64EC)
#  define ARCHITECTURE_ID "ARM64EC"

# elif defined(_M_X64) || defined(_M_AMD64)
#  define ARCHITECTURE_ID "x64"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# elif defined(_M_ARM64)
#  define ARCHITECTURE_ID "ARM64"

# elif defined(_M_ARM)
#  if _M_ARM == 4
#   define ARCHITECTURE_ID "ARMV4I"
#  elif _M_ARM == 5
#   define ARCHITECTURE_ID "ARMV5I"
#  else
#   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
#  endif

# elif defined(_M_MIPS)
#  define ARCHITECTURE_ID "MIPS"

# elif defined(_M_SH)
#  define ARCHITECTURE_ID "SHx"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__WATCOMC__)
```

```
# if defined(_M_I86)
#  define ARCHITECTURE_ID "I86"

# elif defined(_M_IX86)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
# if defined(__ICCARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__ICCRX__)
#  define ARCHITECTURE_ID "RX"

# elif defined(__ICCRH850__)
#  define ARCHITECTURE_ID "RH850"

# elif defined(__ICCRL78__)
#  define ARCHITECTURE_ID "RL78"

# elif defined(__ICCRISCV__)
#  define ARCHITECTURE_ID "RISCV"

# elif defined(__ICCAVR__)
#  define ARCHITECTURE_ID "AVR"

# elif defined(__ICC430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__ICCV850__)
#  define ARCHITECTURE_ID "V850"

# elif defined(__ICC8051__)
#  define ARCHITECTURE_ID "8051"

# elif defined(__ICCSTM8__)
#  define ARCHITECTURE_ID "STM8"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__ghs__)
# if defined(__PPC64__)
#  define ARCHITECTURE_ID "PPC64"

# elif defined(__ppc__)
#  define ARCHITECTURE_ID "PPC"

# elif defined(__ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__x86_64__)
#  define ARCHITECTURE_ID "x64"

# elif defined(__i386__)
#  define ARCHITECTURE_ID "X86"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#elif defined(__TI_COMPILER_VERSION__)
# if defined(__TI_ARM__)
#  define ARCHITECTURE_ID "ARM"

# elif defined(__MSP430__)
#  define ARCHITECTURE_ID "MSP430"

# elif defined(__TMS320C28XX__)
#  define ARCHITECTURE_ID "TMS320C28x"

# elif defined(__TMS320C6X__) || defined(_TMS320C6X)
#  define ARCHITECTURE_ID "TMS320C6x"

# else /* unknown architecture */
#  define ARCHITECTURE_ID ""
# endif

#else
#  define ARCHITECTURE_ID
#endif

/* Convert integer to decimal digit literals.  */
```

```
#define DEC(n)                   \
  (’0’ + (((n) / 10000000)%10)), \
  (’0’ + (((n) / 1000000)%10)),  \
  (’0’ + (((n) / 100000)%10)),   \
  (’0’ + (((n) / 10000)%10)),    \
  (’0’ + (((n) / 1000)%10)),     \
  (’0’ + (((n) / 100)%10)),      \
  (’0’ + (((n) / 10)%10)),       \
  (’0’ +  ((n) % 10))

/* Convert integer to hex digit literals.  */
#define HEX(n)             \
  (’0’ + ((n)»28 & 0xF)), \
  (’0’ + ((n)»24 & 0xF)), \
  (’0’ + ((n)»20 & 0xF)), \
  (’0’ + ((n)»16 & 0xF)), \
  (’0’ + ((n)»12 & 0xF)), \
  (’0’ + ((n)»8  & 0xF)), \
  (’0’ + ((n)»4  & 0xF)), \
  (’0’ + ((n)     & 0xF))

/* Construct a string literal encoding the version number components. */
#ifdef COMPILER_VERSION_MAJOR
char const info_version[] = {
  ’I’, ’N’, ’F’, ’O’, ’:’,
  ’c’,’o’,’m’,’p’,’i’,’l’,’e’,’r’,’_’,’v’,’e’,’r’,’s’,’i’,’o’,’n’,’[’,
  COMPILER_VERSION_MAJOR,
# ifdef COMPILER_VERSION_MINOR
  ’.’, COMPILER_VERSION_MINOR,
#  ifdef COMPILER_VERSION_PATCH
   ’.’, COMPILER_VERSION_PATCH,
#   ifdef COMPILER_VERSION_TWEAK
    ’.’, COMPILER_VERSION_TWEAK,
#   endif
#  endif
# endif
  ’]’,’\0’};
#endif

/* Construct a string literal encoding the internal version number. */
#ifdef COMPILER_VERSION_INTERNAL
char const info_version_internal[] = {
  ’I’, ’N’, ’F’, ’O’, ’:’,
  ’c’,’o’,’m’,’p’,’i’,’l’,’e’,’r’,’_’,’v’,’e’,’r’,’s’,’i’,’o’,’n’,’_’,
  ’i’,’n’,’t’,’e’,’r’,’n’,’a’,’l’,’[’,
  COMPILER_VERSION_INTERNAL,’]’,’\0’};
#endif

/* Construct a string literal encoding the version number components. */
#ifdef SIMULATE_VERSION_MAJOR
char const info_simulate_version[] = {
  ’I’, ’N’, ’F’, ’O’, ’:’,
  ’s’,’i’,’m’,’u’,’l’,’a’,’t’,’e’,’_’,’v’,’e’,’r’,’s’,’i’,’o’,’n’,’[’,
  SIMULATE_VERSION_MAJOR,
# ifdef SIMULATE_VERSION_MINOR
  ’.’, SIMULATE_VERSION_MINOR,
#  ifdef SIMULATE_VERSION_PATCH
   ’.’, SIMULATE_VERSION_PATCH,
#   ifdef SIMULATE_VERSION_TWEAK
    ’.’, SIMULATE_VERSION_TWEAK,
#   endif
#  endif
# endif
  ’]’,’\0’};
#endif

/* Construct the string literal in pieces to prevent the source from
   getting matched.  Store it in a pointer rather than an array
   because some compilers will just produce instructions to fill the
   array rather than assigning a pointer to a static array.  */
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";



#if defined(__INTEL_COMPILER) && defined(_MSVC_LANG) && _MSVC_LANG < 201403L
#  if defined(__INTEL_CXX11_MODE__)
#    if defined(__cpp_aggregate_nsdmi)
#      define CXX_STD 201402L
#    else
#      define CXX_STD 201103L
#    endif
#  else
#    define CXX_STD 199711L
#  endif
#elif defined(_MSC_VER) && defined(_MSVC_LANG)
#  define CXX_STD _MSVC_LANG
```

```
#else
#  define CXX_STD __cplusplus
#endif

const char* info_language_dialect_default = "INFO" ":" "dialect_default["
#if CXX_STD > 202002L
  "23"
#elif CXX_STD > 201703L
  "20"
#elif CXX_STD >= 201703L
  "17"
#elif CXX_STD >= 201402L
  "14"
#elif CXX_STD >= 201103L
  "11"
#else
  "98"
#endif
"]";

/*--------------------------------------------------------------------------*/

int main(int argc, char* argv[])
{
  int require = 0;
  require += info_compiler[argc];
  require += info_platform[argc];
#ifdef COMPILER_VERSION_MAJOR
  require += info_version[argc];
#endif
#ifdef COMPILER_VERSION_INTERNAL
  require += info_version_internal[argc];
#endif
#ifdef SIMULATE_ID
  require += info_simulate[argc];
#endif
#ifdef SIMULATE_VERSION_MAJOR
  require += info_simulate_version[argc];
#endif
#if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
  require += info_cray[argc];
#endif
  require += info_language_dialect_default[argc];
  (void)argv;
  return require;
}
```

6.9 out/build/x64-Debug/CMakeFiles/ShowIncludes/foo.h File Reference

6.10 foo.h

Go to the documentation of this file.

```
```

6.11 out/build/x64-Debug/CMakeFiles/ShowIncludes/main.c File Reference

```
#include "foo.h"
```

Functions

- int main ()

6.11.1 Function Documentation

6.11.1.1 main()

```
int main ( )
```

Definition at line 2 of file main.c.

6.12 main.c

Go to the documentation of this file.

```
00001 #include "foo.h"
00002 int main(){}
```

6.13 String.cpp File Reference

```
#include "String.h"
```

6.14 String.cpp

Go to the documentation of this file.

```
#include "String.h"
#include <string.h>

namespace MyString {

    String::String(size_t init_size) {
        this->data = new char[init_size];
        this->size = init_size;

        if (this->data != NULL) {
            for (size_t i = 0; i < init_size; i++) {
                data[i] = '\0';
            }
        }
    }

    String::String(const char* data) {

        if (!data) {//make sure the passed array is not NULL
        size_t size = strlen(data);

        this->data = new char[size];
        this->size = size;


    }

    String::String(const String& string) {
        this->size = string.getLength();
        this->data = new char[this->size];

        if (this->data != NULL) {
            strncpy(this->data, string.data, this->size);
        }
    }

    String::~String() {
```

```
    if (this->data != NULL)
        delete[] this->data;
}


char* String::getText() {
    return data;
}

void String::setText(const char* text) {
    delete[] this->data;

    this->size = strlen(text);
    this->data = new char[this->size];

    strncpy(this->data, text, this->size);
}

size_t String::getLength() const {
    return strlen(this->data);
}

void String::setLength(size_t size) {
    size_t old_length = this->getLength();
    char* old_data = this->data;

    this->size = size;
    this->data = new char[size];

    // Copy the characters (or add \0 if there are no characters anymore).
    for (size_t i = 0; i < size; i++) {
        if (i < old_length) {
            this->data[i] = old_data[i];
        }
        else {
            this->data[i] = '\0';
        }
    }

    // Destroy the old data
    delete[] old_data;

    // Ensure that the last element is \0
    this->data[size] = '\0';
}

void String::add(const String& text) {
    size_t new_size = this->size + text.size;
    setLength(new_size);

    size_t length = getLength();
    for (size_t i = length; i < new_size; i++) {
        this->data[i] = text.data[i - length];
    }
}

char String::get(size_t pos) const {
    if (pos > getLength()) {
        return '\0';
    }
    return data[pos];
}

bool String::compare(const String& string) const {
    return strcmp(this->data, string.data) == 0;
}

char String::operator[](size_t pos) {
    return this->get(pos);
}


String operator+(const String& string1, const String& string2) {
    String string = String(string1);
    string.add(string2);

    return string;
}

bool operator==(const String& string1, const String& string2) {
    return string1.compare(string2);
}


std::ostream& operator«(std::ostream& ostream, const MyString::String& string) {
    return (ostream « string.data);
}
```

```
}
```

6.15 String.h File Reference

`#include <iostream>`

Classes

- class MyString::It< T >
- class MyString::String

Namespaces

- namespace MyString

Functions

- String MyString::operator+ (const String &string1, const String &string2)
- bool MyString::operator== (const String &string1, const String &string2)
- std::ostream & MyString::operator<< (std::ostream &iostream, const MyString::String &string)

6.16 String.h

Go to the documentation of this file.

```
#ifndef _H_MyString
#define _H_MyString

#include <iostream>

namespace MyString {

    template<typename T>
    class It {
    private:
        T* data;
    public:
        It(T* data) {
            this->data = data;
        }

        T& operator*() {
            return *data;
        }

        It<T>& operator++() {
            data++;
            return *this;
        }

        bool operator==(const It<T>& a) {
            return data == a.data;
        }

        bool operator!=(const It<T>& a) {
            return data != a.data;
        }

    };

    class String {
        friend std::ostream& operator«(std::ostream& iostream, const MyString::String& string);

    private:
        char* data;
        size_t size;
    public:
        String(const char* data);

        String(size_t init_size = 20);

        String(const String& string);

        ~String();

        char* getText();

        void setText(const char* text);

        size_t getLength() const;

        void setLength(size_t size);

        void add(const String& text);

        char get(size_t pos) const;

        bool compare(const String& string) const;

        char operator[](size_t pos);

        It<char> begin() {
            It<char> it(data);
            return it;
        }

        It<char> end()
        {
            It<char> it(data + size);
            return it;
        }
    };

    String operator+(const String& string1, const String& string2);

    bool operator==(const String& string1, const String& string2);

    std::ostream& operator«(std::ostream& iostream, const MyString::String& string);
}
#endif
```

6.17 utilstring.cpp File Reference

```
#include "utilstring.h"
```

Namespaces

- namespace util

Functions

- std::ostream & util::operator<< (std::ostream &iostream, const util::string &myString)
- bool util::operator== (const std::string &lhsString, const util::string &rhsString)
- bool util::operator== (const char ∗lhsCharArray, const util::string &rhsString)
- bool util::operator!= (const std::string &lhsString, const util::string &rhsString)
- bool util::operator!= (const char ∗lhsCharArray, const util::string &rhsString)
- void util::concat (char ∗rawCharTarget, char ∗rawCharSource, size_t startPosition)
- void util::deepCopy (char ∗rawCharTarget, const char ∗rawCharSource, size_t destStartPosition, size_t src←EndPosition)
- void util::printHeader (const char ∗text)
- void util::printSubHeader (const char ∗text)

6.17.1 Detailed Description

===

Author

: Nour Ahmed @email : nahmed@stud.hs-bremen.de, nourbrm02@gmail.com @repo ↩
: https://github.com/nouremara/cpp_mystring @createdOn : 23.11.2022 @description
: implementation of own string class

this file presents an implementation of a class named string. This class behavior will be similar to the std::string and both

6.17.2 std::string and this string class are compatible.

Definition in file utilstring.cpp.

6.18 utilstring.cpp

Go to the documentation of this file.

```
// Nour Ahmed
// Matrikal-Nr.: 5200991
// Assignment 1 - Own String Class

#include "utilstring.h"

namespace util {
    /*=====================================================================*
     *                          Constructors                               *
     *=====================================================================*/

    string::string(size_t intialSize) {
        intialize_string(intialSize);
    }
    //-----------------------------------------------------------------


    string::string(const char* data) {
        intialize_string(rawSize(data));  // ensure string is initialized before using it
        deepCopy(data);     // copy passed array to the string
    }
    //-----------------------------------------------------------------

    string::string(const std::string& data) {
        intialize_string(rawSize(data.c_str()));  // ensure string is initialized before using it
        deepCopy(data.c_str());     // copy passed array to the string
    }
    //-----------------------------------------------------------------

    string::string(const string& data) {
        intialize_string(rawSize(data.c_str()));  // ensure string is initialized before using it
        deepCopy(data.c_str());        // copy passed array to the string
    }


    /*=====================================================================*
     *                          Destructor                                 *
     *=====================================================================*/
    string::~string(void) {
        delete[] internal_buffer;
    }


    /*=====================================================================*
     *                          Methods                                    *
     *=====================================================================*/

    void string::intialize_string(size_t length) {
        internal_buffer = new char[length + 1];
        buffer_size = length + 1;

        // initialize an empty string
        internal_buffer[0] = '\0';
    }
    //------------------------------------------------------------

    size_t string::size() const {
        return rawSize(internal_buffer);
    }

    size_t string::length() const {
        return rawSize(internal_buffer);
    }
    //------------------------------------------------------------

    size_t string::rawSize(const char* rawChar) {
        size_t length = 0;
        while (rawChar[length] != '\0') {
            length++;
        }

        return length;
    }
    //------------------------------------------------------------


    void string::deepCopy(const char* rawChar, size_t startPosition) {
        //check if internalData is of enough size to accommodate the passed array
        size_t rawCharSize = rawSize(rawChar);
        if (rawCharSize > size()) { // more space is needed
            //delete current internalData
            delete[] internal_buffer;

            // re-initialize the string with the required size
            intialize_string(rawCharSize);
        }

        //copy the passed array to the newly allocated internalData
        int j = startPosition;
        while (rawChar[j] != '\0') {
            internal_buffer[j] = rawChar[j];
            j++;
        }

        internal_buffer[j] = '\0';      // ensure destination string is null terminated
        //string_size = rawCharSize;        // set string size to the new one
    }
    //------------------------------------------------------------


    char* string::c_str() const {
        return internal_buffer;
    }
    //------------------------------------------------------------


    // Returns a pointer to an array that contains a null-terminated
        // sequence of characters(i.e., a C-string) representing the current
        // value of the string object.
        // Returns a substring object of type util::string which
        // starts at pos. Parameter length specifies the amount of
        // characters of the new util::string to be returned.
    string string::substr(size_t pos, size_t length) {
        string substring(length);

        util::deepCopy(substring.c_str(), internal_buffer+ pos, 0, length);


        return substring;
    }

    //------------------------------------------------------------

    int string::compare(const char* lhsCharArray, const char* rhsCharArray)
    {
        // convert both pointers from 'char*' to 'unsigned char*'
        // needed for the difference calculations
        const unsigned char* p1 = (const unsigned char*)lhsCharArray;
        const unsigned char* p2 = (const unsigned char*)rhsCharArray;

        // check if characters differ, or end of the first string (a terminating null) is reached
        while (*p1 && *p1 == *p2) {
            // proceed to the next pair of characters
            ++p1, ++p2;
        }

        // return the ASCII difference
        return (*p1 > *p2) - (*p2 > *p1);
    }


    /*=========================================================================*
     *                          Operators                                      *
     *=========================================================================*/



     //------------------------------------------------------------

    string& string::operator=(const string& rhsString) {
        deepCopy(rhsString.c_str());
        return *this;
    }

    string& string::operator=(const char* rhsCharArray) {
        deepCopy(rhsCharArray);
        return *this;
    }

    string& string::operator=(const std::string& rhsString) {
        deepCopy(rhsString.c_str());
        return *this;
    }
    //------------------------------------------------------------


    bool string::operator==(const string& rhsString) {
        //if (string_size != rhsString.size()) return false;
        // note that compare returns 0 when the two strings are equal
        return  !compare(internal_buffer, rhsString.c_str());
    }

    bool string::operator==(const std::string& rhsString) {
        //if (string_size != rhsString.size()) return false;

        // note that compare returns 0 when the two strings are equal
        return  !compare(internal_buffer, rhsString.c_str());
    }

    bool string::operator==(const char* charArray) {
        //if (string_size != rhsString.size()) return false;

        // note that compare returns 0 when the two strings are equal
        return  !compare(internal_buffer, charArray);
    }
    //------------------------------------------------------------

    bool string::operator!=(const string& rhsString) {
        //if (string_size != rhsString.size()) return false;
        // note that compare returns 0 when the two strings are equal
        return  compare(internal_buffer, rhsString.c_str());
    }

    bool string::operator!=(const std::string& rhsString) {
        //if (string_size != rhsString.size()) return false;

        // note that compare returns 0 when the two strings are equal
        return  compare(internal_buffer, rhsString.c_str());
    }

    bool string::operator!=(const char* charArray) {
        //if (string_size != rhsString.size()) return false;

        // note that compare returns 0 when the two strings are equal
        return  compare(internal_buffer, charArray);
    }
    //------------------------------------------------------------

    const char string::operator[](size_t position) {
        if (position > size()) return '\0';
        return internal_buffer[position];
    }
    //------------------------------------------------------------


    string& string::operator+(const string& rhsString) {
        size_t total_size = size() + rhsString.size() + 1;
        char* temp = new char[total_size];

        util::deepCopy(temp, internal_buffer,0);
        util::deepCopy(temp, rhsString.c_str(), size());

        delete[] internal_buffer;

        internal_buffer = temp;
        buffer_size = total_size;
        std::cout « "\n\nbuffer_size " « buffer_size « std::endl;
        std::cout « "(size: " « size() « ") : " « internal_buffer « std::endl;

        return *this;
    }
    //------------------------------------------------------------------


    string& string::operator+(const char* rhsString) {


        // a temporary object to fill it with the concatenated strings
        // total size of the two strings combined
        size_t rhsSize = rawSize(rhsString);
        char* tempData = new char[size() + rhsSize + 1];

        // copy the passed array to the newly allocated tempData
        // copy the lhs string from the beginning of the string then the rhs string there after
        // ------------------------------------------------------------
        // |0| ... |lhs size| ...  |lhs size|rhs size + lhs size + 1|
        // |   lhs string   |   rhs string  | \0 |
        // ------------------------------------------------------------
        // note that at the given position the lhs \0 termination will be overwritten as this copy starts at its position

        int j = 0;
        while (internal_buffer[j] != '\0') {
            tempData[j] = internal_buffer[j];
            j++;
        }

        int i = 0;
        while (rhsString[i] != '\0') {
            tempData[j] = rhsString[i];
            j++;
            i++;
        }
        tempData[j] = '\0'; // ensure destination string is null terminated

        std::cout « " > [tempData] : " « tempData « std::endl;

        util::string temp;
        //delete[] tempData;

        return temp;
    }



    /*========================================================================================*
     * non-member functions and operator methods for the cases util::string is on the rhs *
     *========================================================================================*/

    std::ostream& operator«(std::ostream& iostream, const util::string& myString) {
        return (iostream « myString.c_str());
    }

    //------------------------------------------------------------------


    bool operator==(const std::string& lhsString, const util::string& rhsString) {
        // note that compare returns 0 when the two strings are equal
        return  !util::string::compare(lhsString.c_str(), rhsString.c_str());
    }

    bool operator==(const char* lhsCharArray, const util::string& rhsString) {
        // note that compare returns 0 when the two strings are equal
        return  !util::string::compare(lhsCharArray, rhsString.c_str());
    }
    //------------------------------------------------------------------

    bool operator!=(const std::string& lhsString, const util::string& rhsString) {
        // note that compare returns 0 when the two strings are equal
        return  util::string::compare(lhsString.c_str(), rhsString.c_str());
    }

    bool operator!=(const char* lhsCharArray, const util::string& rhsString) {
        // note that compare returns 0 when the two strings are equal
        return  util::string::compare(lhsCharArray, rhsString.c_str());
    }
    //------------------------------------------------------------------


    void concat(char* rawCharTarget, char* rawCharSource, size_t startPosition) {
        // if the startPosition is not given (we have its default -1)
        // do a normal concatenation of the two strings)
        // note that at the given position the lhs \0 termination will be
        // overwritten as this copy starts at its position
        if (startPosition == -1) {
            startPosition = util::string::rawSize(rawCharTarget);
        }

        // deep copy rawCharSource into rawCharTarget beginning at startPosition
        for (size_t j = 0; rawCharSource[j] != '\0'; ++j, ++startPosition) {
            rawCharTarget[startPosition] = rawCharSource[j];
        }

        // ensure destination string is null terminated
        rawCharTarget[startPosition] = '\0';
    }


    // destStartPosition  default is to first location of the destRawChar
    // srcEndPosition     default is to last character (before the \0) of the srcRawChar
    void deepCopy(char* rawCharTarget, const char* rawCharSource, size_t destStartPosition, size_t srcEndPosition){
        // check and adjust for default values
        destStartPosition = (destStartPosition == -1) ? util::string::rawSize(rawCharTarget) : destStartPosition;
        srcEndPosition = (srcEndPosition == -1) ? util::string::rawSize(rawCharSource): srcEndPosition;

        // deep copy rawCharSource into rawCharTarget beginning at startPosition
//      for (size_t j = 0; rawCharSource[j] != '\0'; ++j, ++destStartPosition) {
        for (size_t j = 0; j < srcEndPosition; ++j, ++destStartPosition) {
            rawCharTarget[destStartPosition] = rawCharSource[j];
        }

        // ensure destination string is null terminated
        rawCharTarget[destStartPosition] = '\0';

    }
    //--------------------------------------------------
    void printHeader(const char* text){
        size_t spaces_needed = (80 - util::string::rawSize(text))/2 -2;

        std::cout « "\033[96m"; // set text and background colors
        std::cout « "--------------------------------------------------------------------------------\n-";
        for (int i = 0; i < spaces_needed; ++i) { std::cout « " "; }
        std::cout « text;
        for (int i = 0; i < spaces_needed; ++i) { std::cout « " "; }
        std::cout « " -\n";
        std::cout « "--------------------------------------------------------------------------------\n";
        std::cout « "\033[0m"; // reset text and background colors

    }

    void printSubHeader(const char* text) {
        std::cout « "\033[32m"; // set text and background colors
        std::cout « text;
        std::cout « "\033[0m\n"; // reset text and background colors
    }


} // namespace util
```

6.19 utilstring.h File Reference

implementation of own string class.

```
#include <cstddef>
#include <iostream>
#include <string>
```

Classes

- class util::string

Namespaces

- namespace util

Macros

- #define INITIAL_SIZE 10

Functions

- void util::concat (char ∗rawCharTarget, char ∗rawCharSource, size_t startPosition)
- void util::deepCopy (char ∗rawCharTarget, const char ∗rawCharSource, size_t destStartPosition, size_t src←EndPosition)
- void util::printHeader (const char ∗text)
- void util::printSubHeader (const char ∗text)

6.19.1 Detailed Description

implementation of own string class.

==

this file presents an implementation of a class named string. This class behavior will be similar and compatible to the std::string. This file contains the prototypes for the class, its methods and eventually any macros, constants, or global variables you will need to use it.

Author

: Nour Ahmed @email : `nahmed@stud.hs-bremen.de`, nour @repo : `https://github.←com/nouremara/cpp_mystring`

6.19.2 @createdOn : 23.11.2022

Definition in file utilstring.h.

6.19.3 Macro Definition Documentation

6.19.3.1 INITIAL_SIZE

```
#define INITIAL_SIZE 10
```

Definition at line 33 of file utilstring.h.

6.20 utilstring.h

Go to the documentation of this file.

```
// Nour Ahmed
// Matrikal-Nr.: 5200991
// Assignment 1 - Own string Class

// see: https://gist.github.com/philipheimboeck/099e540d800063e3e6ec
// see: https://codereview.stackexchange.com/questions/98329/stdstring-implementation
// see: https://en.wikipedia.org/wiki/Snake_case

#ifndef UTILSTRING_H
#define UTILSTRING_H

#include <cstddef>
#include <iostream>
#include <string>

namespace util {
    // Initially, the class shall provide memory for 10 printable characters
    #define INITIAL_SIZE 10

    class string {
        char* internal_buffer;
        size_t buffer_size;
        //size_t string_size;

       public:
        /*================== Constructors ===============*/
        string(size_t intialSize= INITIAL_SIZE);
        string(const char*);
        string(const std::string&);
        string(const string&);
        /*================== Destructor =================*/
        ~string(void);


        /*================== Methods ====================*/

        void intialize_string(size_t length = 0);
        void deepCopy(const char* rawChar, size_t startPosition = 0);
        //void concat(char* rawCharTarget, const char* rawCharSource, size_t startPosition = 0);



        // Returns a pointer to an array that contains a null-terminated
        // sequence of characters(i.e., a C-string) representing the current
        // value of the string object.
        // Returns a substring object of type util::string which
        // starts at pos. Parameter length specifies the amount of
        // characters of the new util::string to be returned.
        string substr(size_t pos, size_t length);

        // Allows raw access to the internal C-string (through its char* pointer)
        char* c_str() const;

        bool compare(const char* charArray) const;
        static int compare(const char* s1, const char* s2);

        // Clears your string object
        // Erases the contents of the string, which becomes an empty string(with
        // a length of 0 characters).
        void clear();


        size_t size() const;
        static size_t rawSize(const char* rawChar);

        // Returns the amount of characters of your string excluding \0.
        // Might be smaller than the actual reserved memory.
        size_t length() const;


        /*================== Operators ====================*/
        string& operator+(const string& rhsString);
        string& operator+(const char* strInstance);


        string& operator=(const string& rhsString);
        string& operator=(const char* rhsCharArray);
        string& operator=(const std::string& rhsString);

        bool operator==(const string& rhsString);
        bool operator==(const std::string& rhsString);
        bool operator==(const char* charArray);

        bool operator!=(const string& rhsString);
        bool operator!=(const std::string& rhsString);
        bool operator!=(const char* charArray);


        const char operator[](size_t position);


        /*=========================================================================*
         *              Non-member function overloads                              *
         *=========================================================================*/
         // Free operator methods for the cases util::string is on the rhs
         // Friendship enables access to private members
        friend std::ostream& operator«(std::ostream& iostream, const util::string& myString);

        friend bool operator==(const std::string& lhsString, const util::string& rhsString);
        friend bool operator==(const char* lhsCharArray, const util::string& rhsString);

        friend bool operator!=(const std::string& lhsString, const util::string& rhsString);
        friend bool operator!=(const char* lhsCharArray, const util::string& rhsString);
    };

    /*=========================================================================*
     *              Some Utility functions                                     *
     *=========================================================================*/
    void concat(char* rawCharTarget,  char* rawCharSource, size_t startPosition = -1);

    void deepCopy(char* rawCharTarget, const char* rawCharSource, size_t destStartPosition = -1,
 size_t srcEndPosition = -1);



//-------------------------------------------------
    void printHeader(const char* text);
    void printSubHeader(const char* text);



}  // namespace util


#endif /* UTILSTRING_H */
```