Articles » Languages » C / C++ Language » General

C++: Minimalistic CSV Streams

Shao Voon Wong, 25 Aug 2016 MIT

4.77 (58 votes)

Read/write CSV in few lines of code!

Download minicsv-1.8.0.zip - 14.5 KB See breaking change in History section

Introduction

MiniCSV is a small, single header library which is based on C++ file streams and is comparatively easy to use. Without further ado, let us see some code in action.

Writing

We see an example of writing tab-separated values to file using `csv::ofstream` class. Now you can specify the escape string when calling `set_delimiter` in version 1.7

```
#include "minicsv.h"

struct Product
{
    Product() : name(""), qty(0), price(0.0f) {}
    Product(std::string name_, int qty_, float price_)
        : name(name_), qty(qty_), price(price_) {}
    std::string name;
    int qty;
    float price;
};

int main()
{
    csv::ofstream os("products.txt", std::ios_base::out);
    os.set_delimiter('\t', "##");
    if(os.is_open())
    {
        Product product("Shampoo", 200, 15.0f);
        os << product.name << product.qty << product.price << NEWLINE;
        Product product2("Soap", 300, 6.0f);
```

```
        os << product2.name << product2.qty << product2.price << NEWLINE;
    }
    os.flush();
    return 0;
}
```

`NEWLINE` is defined as `'\n'`. We cannot use `std::endl` here because `csv::ofstream` is not derived from the `std::ofstream`.

Reading

To read back the same file, `csv::ifstream` is used and `std::cout` is for displaying the read items on the console.

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

int main()
{
    csv::ifstream is("products.txt", std::ios_base::in);
    is.set_delimiter('\t', "##");
    if(is.is_open())
    {
        Product temp;
        while(is.read_line())
        {
            is >> temp.name >> temp.qty >> temp.price;
            // display the read items
            std::cout << temp.name << "," << temp.qty << "," << temp.price
<< std::endl;
        }
    }
    return 0;
}
```

The output in console is as follows.

```
Shampoo,200,15
Soap,300,6
```

Overloaded stream operators

String stream has been introduced in v1.6. Let me show you an example on how to overload string stream operators for the Product class. The concept is the same for file streams.

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

struct Product
{
    Product() : name(""), qty(0), price(0.0f) {}
    Product(std::string name_, int qty_, float price_) : name(name_),
qty(qty_), price(price_) {}
    std::string name;
    int qty;
    float price;
};
```

```
template<>
inline csv::istringstream& operator >> (csv::istringstream& istm, Product&
val)
{
    return istm >> val.name >> val.qty >> val.price;
}

template<>
inline csv::ostringstream& operator << (csv::ostringstream& ostm, const
Product& val)
{
    return ostm << val.name << val.qty << val.price;
}

int main()
{
    // test string streams using overloaded stream operators for Product
    {
        csv::ostringstream os;
        os.set_delimiter(',', "$$");
        Product product("Shampoo", 200, 15.0f);
        os << product << NEWLINE;
        Product product2("Towel, Soap, Shower Foam", 300, 6.0f);
        os << product2 << NEWLINE;

        csv::istringstream is(os.get_text().c_str());
        is.set_delimiter(',', "$$");
        Product prod;
        while (is.read_line())
        {
            is >> prod;
            // display the read items
            std::cout << prod.name << "|" << prod.qty << "|" << prod.price
<< std::endl;
        }
    }
    return 0;
}
```

This is what is displayed on the console.

```
Shampoo|200|15
Towel, Soap, Shower Foam|300|6
```

What if the type has private members? Create a member function that takes in the stream object.

```
class Product
{
public:
    void read(csv::istringstream& istm)
    {
        istm >> this->name >> this->qty >> this->price;
    }
};

template<>
inline csv::istringstream& operator >> (csv::istringstream& istm, Product&
prod)
{
    prod.read(istm);
    return istm;
}
```

Conclusion

MiniCSV is a small CSV library that is based on C++ file streams. Because delimiter can be changed on the fly, I have used this library to write file parser for MTL and Wavefront OBJ format in a relatively short time compared to handwritten with no library help. MiniCSV is now hosted at Github. Thank you for reading!

History

- **2014-03-09:** Initial Release
- **2014-08-20:** Remove the use of smart ptr
- **2015-03-23:** 75% perf increase on writing by removing the flush on every line, fixed the lnk2005 error of multiple redefinition. `read_line` replace `eof` on `ifstream`.
- **2015-09-22:** v1.7: Escape/unescape and surround/trim quotes on text
- **2015-09-24:** Added overloaded `stringstream` operators example.
- **2015-09-27:** Stream operator overload for `const char*` in v1.7.2.
- **2015-10-04:** Fixed G++ and Clang++ compilation errors in v1.7.3.
- **2015-10-20:** Ignore delimiters within quotes during reading when `enable_trim_quote_on_str` is enabled in v1.7.6. Example: 10.0,"Bottle,Cup,Teaspoon",123.0 will be read as as 3 tokens : <10.0><Bottle,Cup,Teaspoon><123.0>
- **2016-05-05:** Now the quote inside your quoted string are escaped now. Default escape string is `"""` which can be changed through `os.enable_surround_quote_on_str()` and `is.enable_trim_quote_on_str()`
- **2016-07-10:** Version 1.7.9: Reading UTF-8 BOM
- **2016-08-02:** Version 1.7.10 : Separator class for the stream, so that no need to call `set_delimiter` repeatedly if delimiter keep changing. See code example below.

```
// demo sep class usage
csv::istringstream is("vt:33,44,66");
is.set_delimiter(',', "$$");
csv::sep colon(':', "<colon>");
csv::sep comma(',', "<comma>");
while (is.read_line())
{
    std::string type;
    int r = 0, b = 0, g = 0;
    is >> colon >> type >> comma >> r >> b >> g;
    // display the read items
    std::cout << type << "|" << r << "|" << b << "|" << g << std::endl;
}
```

- **2016-08-23:** Version 1.7.11 : Fixed `num_of_delimiter` function: do not count delimiter within quotes
- **2016-08-26:** Version 1.8.0 : Added better error message for data conversion during reading. Before that, data conversion error with `std::istringstream` went undetected.
  Before change

```
template<typename T>
csv::ifstream& operator >> (csv::ifstream& istm, T& val)
{
    std::string str = istm.get_delimited_str();

#ifdef USE_BOOST_LEXICAL_CAST
```

```
    val = boost::lexical_cast<T>(str);
#else
    std::istringstream is(str);
    is >> val;
#endif

    return istm;
}
```

After change

```
template<typename T>
csv::ifstream& operator >> (csv::ifstream& istm, T& val)
{
    std::string str = istm.get_delimited_str();

#ifdef USE_BOOST_LEXICAL_CAST
    try
    {
        val = boost::lexical_cast<T>(str);
    }
    catch (boost::bad_lexical_cast& e)
    {
        throw std::runtime_error(istm.error_line(str).c_str());
    }
#else
    std::istringstream is(str);
    is >> val;
    if (!(bool)is)
    {
        throw std::runtime_error(istm.error_line(str).c_str());
    }
#endif

    return istm;
}
```

Breaking changes: It means old user code to catch `boost::bad_lexical_cast` must be changed to catch `std::runtime_error`. Same for `csv::istringstream`. Beware `std::istringstream` is not as good as `boost::lexical_cast` at catching error. Example, `"4a"` gets converted to integer `4` without error.
Example of the `csv::ifstream` error log as follows

```
csv::ifstream conversion error at line no.:2, filename:products.txt,
token position:3, token:aa
```

Similar for `csv::istringstream` except there is no filename.

```
csv::istringstream conversion error at line no.:2, token position:3,
token:aa
```

Related Articles

- C++: Minimalistic CSV Streams
- C++: Simplistic Binary Streams
- C++: New Text Streams

License

Share

About the Author

Shao Voon Wong

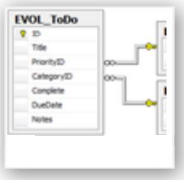
Software Developer (Senior)

United States

IT Certifications

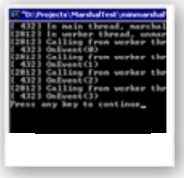
- IT Infrastructure Library Foundational (ITIL v3)
- Scrum Alliance Certified Scrum Master (CSM)
- EC-Council Certified Secure Programmer (ECSP) .NET
- EC-Council Certified Ethical Hacker (CEH)
- EC-Council Certified Security Analyst (ECSA)
- Certified Secure Software Lifecycle Professional (CSSLP)

You may also be interested in...



Minimalist Meta-Model for CRUD Applications

An Introduction to Application Performance Management (APM)



Minimalist In-Process Interface Marshaling

Microsoft Data Science Virtual Machine for Windows and Linux now

Comments and Discussions

26 messages have been posted for this article Visit **http://www.codeproject.com/Articles/741183/Cplusplus-Minimalistic-CSV-Streams** to post and view comments on this article, or click **here** to get a print view with messages.