Martin Broadhurst

HOW TO SPLIT A STRING IN C++

JULY 30, 2016 | MARTIN | 1 COMMENT

Java has `String.split()`, Python has `string.split()`, Perl has `split`. There is no simple string-splitting method in C++, but there are plenty of ways of doing it. Here are some methods:

1. Put it in a `stringstream` and extract the tokens
2. Put it in a `stringstream` and use `getline()` with a delimiter
3. Use `string::find` progressively
4. Use `string::find_first_of` progressively with a number of delimiters
5. Use `boost::split()`
6. Use `boost::split_iterator`
7. Use `boost::tokenizer`
8. Use `boost::sregex_token_iterator`
9. Use `pystring::split`
10. Use my C `split` function

1. Put it in a `stringstream` and extract the tokens

```
#include <string>
#include <sstream>
#include <algorithm>
#include <iterator>

template <class Container>
void split1(const std::string& str, Container& cont)
{
    std::istringstream iss(str);
    std::copy(std::istream_iterator<std::string>(iss),
         std::istream_iterator<std::string>(),
         std::back_inserter(cont));
}
```

2. Put it in a `stringstream` and use `getline()` with a delimiter

```
#include <string>
```

```
#include <sstream>
#include <algorithm>
#include <iterator>

template <class Container>
void split2(const std::string& str, Container& cont, char delim = ' '
{
    std::stringstream ss(str);
    std::string token;
    while (std::getline(ss, token, delim)) {
        cont.push_back(token);
    }
}
```

3. Use `string::find` progressively

```
#include <string>
#include <algorithm>
#include <iterator>

template <class Container>
void split3(const std::string& str, Container& cont,
                char delim = ' ')
{
    std::size_t current, previous = 0;
    current = str.find(delim);
    while (current != std::string::npos) {
        cont.push_back(str.substr(previous, current - previous));
        previous = current + 1;
        current = str.find(delim, previous);
    }
    cont.push_back(str.substr(previous, current - previous));
}
```

4. Use `string::find_first_of` progressively with a number of delimiters

```
#include <string>
#include <algorithm>
#include <iterator>

template <class Container>
void split4(const std::string& str, Container& cont,
                const std::string& delims = " ")
{
    std::size_t current, previous = 0;
    current = str.find_first_of(delims);
    while (current != std::string::npos) {
        cont.push_back(str.substr(previous, current - previous));
        previous = current + 1;
        current = str.find_first_of(delims, previous);
    }
    cont.push_back(str.substr(previous, current - previous));
}
```

5. Use `boost::split()`

```
#include <string>
#include <boost/algorithm/string.hpp>

template <class Container>
void split5(const std::string& str, Container& cont,
              const std::string& delims = " ")
{
    boost::split(cont, str, boost::is_any_of(delims));
}
```

Reference: Function template split

6. Use `boost::split_iterator`

```
#include <string>
#include <boost/algorithm/string.hpp>

template <class Container>
void split6(const std::string& str, Container& cont,
              char delim = ' ')
{
    typedef boost::split_iterator<std::string::const_iterator> splite
    std::string sdelim(1, delim);
    for (spliterator it = boost::make_split_iterator(str,
              boost::first_finder(sdelim, boost::is_equal()));
              it != spliterator(); ++it) {
        cont.push_back(boost::copy_range<std::string>(*it));
    }
}
```

Reference: Function template make_split_iterator

7. Use `Use boost::tokenizer`

```
#include <string>
#include <algorithm>
#include <boost/tokenizer.hpp>

template <class Container>
void split7(const std::string& str, Container& cont,
              const std::string& delims = " ")
{
    typedef boost::char_separator<char> separator;
    boost::tokenizer<separator> tokens(str, separator(delims.c_str())
    std::copy(tokens.begin(), tokens.end(), std::back_inserter(cont))
}
```

Reference: Tokenizer Class

8. Use `boost::sregex_token_iterator`

```
#include <string>
#include <algorithm>
#include <boost/regex.hpp>

template <class Container>
void split8(const std::string& str, Container& cont,
                const std::string delim = "\\s+")
{
    boost::regex re(delim);
    std::copy(boost::sregex_token_iterator(str.begin(), str.end(), re
            boost::sregex_token_iterator(),
            std::back_inserter(cont));
}
```

Reference: regex_token_iterator

9. Use `pystring::split()`

```
#include <pystring.h>

template <class Container>
void split9(const std::string& str, Container& cont,
                const std::string delim = " ")
{
    std::vector<std::string> vec;
    pystring::split(str, vec, delim);
    std::copy(vec.begin(), vec.end(), std::back_inserter(cont));
}
```

Reference: pystring/pystring.h

10. Use my C split function

```
template <class Container>
void add_to_container(const char *str, size_t len, void *data)
{
    Container *cont = static_cast<Container*>(data);
    cont->push_back(std::string(str, len));
}

template <class Container>
void split10(const std::string& str, Container& cont, char delim = '
{
    split(str.c_str(), delim, static_cast<split_fn>(add_to_container<
}
```

Reference: Split a string in C

An example program

```cpp
#include <iostream>
#include <string>
#include <algorithm>
#include <iterator>
#include <vector>

int main()
{
    char str[] = "The quick brown fox jumps over the lazy dog";
    std::vector<std::string> words;
    split1(str, words);
    std::copy(words.begin(), words.end(),
         std::ostream_iterator<std::string>(std::cout, "\n"));
}
```

```
The
quick
brown
fox
jumps
over
the
lazy
dog
```

Related

- How to find a substring in C++
- How to do string formatting in C++
- How to replace all occurrences of a character in a std::string
- How to do case-insensitive string comparison in C++
- How to concatenate a string and an int in C++
- How to convert a string to lower or upper case in C++
- How to trim a std::string in C++
- How to get a const char* or a char* from a std::string
- How to convert an int to a std::string in C++

BOOST C++ PYSTRING REGEX SPLIT STRINGS

ONE THOUGHT ON “HOW TO SPLIT A STRING IN C++”