suplibxx

1.3.7

Generated by Doxygen 1.5.6

Thu Oct 2 20:49:06 2008

Contents

Chapter 1

The C++ suplib Library

The suplib C++ library is a collection of groovy, general purpose routines. It's split up into several sub-packages (see the **Modules** page for more information).

1.1 Copyright

1.2 Column Name Selection

These routines concern themselves with selecting or excluding column names based on exact matches or Perl regular expression. To use them, ensure that you include the correct header file:

```
#include <suplib++/colselect.h>
```

1.3 Input/Output

These routines concern themselves with input and output. To use them, ensure that you include the correct header file:

```
#include <suplib++/io.h>
```

1.4 String

These are functions which manipulate strings. Ensure that you include the correct header file:

```
#include <suplib++/str.h>
```

Chapter 2

I/O Examples

```
#include "io.h"

int main( int argc, char** argv ) {
```

We begin each example by creating the input stream. Next, we initialize the string we expect suplib::getrecord to return. Finally, we call getrecord and compare the returned string to the expected string.

```
  {
    // Example 1
    strstream strstr;
    strstr <<  "  now, is, the, time  ";
    expected = "  now, is, the, time  ";
    suplib::getrecord( strstr, returned, suplib::READ_PHYS );
    cout << ( returned == expected ? "OK" : "NOT OK" ) << endl;
  }
```

With suplib::READ_PHYS set, we read up until the '\n' or EOF.

```
  {
    // Example 2
    strstream strstr;
    strstr <<  "  now, is, the, time  \n";
    expected = "  now, is, the, time";
    suplib::getrecord( strstr, returned, suplib::READ_PHYS | suplib::STRIP );
    cout << ( returned == expected ? "OK" : "NOT OK" ) << endl;
  }
```

With suplib::READ_PHYS and suplib::STRIP set, we read up until the '\n' and then strip all whitespace from the end of the string.

```
  {
    // Example 3
    strstream strstr;
    strstr <<   "  now, is, the, time  \\  \n"
           <<   "  now, is, the, time  \\  \n"
           <<   "  now, is, the, time  ";
    expected  = "  now, is, the, time  ";
    expected += "  now, is, the, time  ";
    expected += "  now, is, the, time  ";
    suplib::getrecord( strstr, returned, suplib::READ_LOGICAL );
    cout << ( returned == expected ? "OK" : "NOT OK" ) << endl;
  }
```

With suplib::READ_LOGICAL set, we read all three physical lines.

```
  {
    // Example 4
```

```
  strstream strstr;
  strstr   << "  now, is, the, time  \\  \n"
           << "  now, is, the, time  \\  \n"
           << "  now, is, the, time  ";
  expected  = "  now, is, the, time  \\\n";
  expected += "  now, is, the, time  \\\n";
  expected += "  now, is, the, time  ";
  suplib::getrecord( strstr, returned, suplib::READ_LOGICAL | suplib::CLEAN );
  cout << ( returned == expected ? "OK" : "NOT OK" ) << endl;
}
```

With suplib::READ_LOGICAL and suplib::CLEAN set, we read all three physical lines and remove whitespace between the continuation character and the newline character.

```
{
  // Example 5
  strstream strstr;
  strstr  << "  now, is, the, time  \n";
  expected = "  now, is, the, time";
  suplib::getrecord( strstr, returned, suplib::READ_LOGICAL | suplib::STRIP );
  cout << ( returned == expected ? "OK" : "NOT OK" ) << endl;
}
```

With suplib::READ_LOGICAL and suplib::STRIP set, we read the single physical line since there is no continuation character. We then strip the trailing whitespace from the line.

```
{
  // Example 6
  strstream strstr;
  strstr <<   "  now, is, the, time  \\  \n"
         <<   "  now, is, the, time  \\  \n"
         <<   "  now, is, the, time  ";
  expected  = "  now, is, the, time  ";
  expected += "  now, is, the, time  ";
  expected += "  now, is, the, time";
  suplib::getrecord( strstr, returned, suplib::READ_LOGICAL | suplib::STRIP);
  cout << ( returned == expected ? "OK" : "NOT OK" ) << endl;
}
```

With suplib::READ_LOGICAL and suplib::STRIP set, we read the three physical lines. We then strip the continuation character and trailing whitespace from the line.

```
{
  // Example 7
  strstream strstr;
  strstr << "  now, is, the, time  \\  \n"
         << "  now, is, the, time  \\  \n"
         << "  now, is, the, time  ";
  expected  = "  now, is, the, time  \\\n";
  expected += "  now, is, the, time  \\\n";
```

```
  expected += "  now, is, the, time";
  suplib::getrecord( strstr, returned, suplib::READ_LOGICAL | suplib::STRIP | suplib::CLEAN );
  cout << ( returned == expected ? "OK" : "NOT OK" ) << endl;
}
```

With suplib::READ_LOGICAL, suplib::STRIP, and suplib::CLEAN set, we read the three physical lines. We then strip the continuation character and trailing whitespace from the line.

```
{
  // Example 8
  strstream strstr;
  strstr   << "  now, is, the, time  -  \n"
           << "  now, is, the, time  -  \n"
           << "  now, is, the, time  ";
  expected  = "  now, is, the, time  ";
  expected += "  now, is, the, time  ";
  expected += "  now, is, the, time  ";
  suplib::getrecord( strstr, returned, suplib::READ_LOGICAL, '\n', '-' );
  cout << ( returned == expected ? "OK" : "NOT OK" ) << endl;
}
```

Here we change the definition of the continuation character.

```
}
```

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Directory Hierarchy

4.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 6

Module Documentation

6.1 Column Selection

Functions

- void suplib::colselect (const vector< string > &icolumns, const vector< string > &exact_add, const vector< string > &regex_add, const vector< string > &exact_del, const vector< string > &regex_del, vector< string > &ocolumns) throw (Exception)

  select columns based on exact/regex matching/exclusion.

- bool suplib::match (const string &str, const string &pattern) throw (Exception)

  handles Perl regular expression matching.

6.1.1 Function Documentation

6.1.1.1 void suplib::colselect (const vector< string > & *icolumns*, const vector< string > & *exact_add*, const vector< string > & *regex_add*, const vector< string > & *exact_del*, const vector< string > & *regex_del*, vector< string > & *ocolumns*) throw (Exception)

select columns based on exact/regex matching/exclusion.

Parameters:

icolumns vector of column names on which to operate

exact_add column names to add to output

regex_add regular expressions used to add to output

exact_del column names to exclude from output

regex_del regular expressions used to exclude from output

ocolumns set of column names which were selected

Exceptions:

Exception errors related to pattern matching

colselect places strings from the input vector in the output vector based on matches specified by the parameters. Strings appearing in the exact_del parameter are excluded from the output vector. Strings which match the regular expression in regex_del are excluded from the output vector. Strings which do not match any of the ∗_del parameters but are either present in exact_add or match a regular expression in regex_add are added to the output set.

Definition at line 52 of file colselect.cc.

6.1.1.2 bool suplib::match (const string & *str*, const string & *pattern*) throw (Exception)

handles Perl regular expression matching.

Parameters:

str the string

pattern the regular expression pattern with which to compare

Exceptions:

Exception errors related to pattern matching

match uses the pcre library's functionality to compare str with pattern.

Returns:

true is str matches pattern, false if not.

Definition at line 45 of file match.cc.

6.2 Input/Output

Enumerations

- enum suplib::readopt {

  suplib::READ_PHYS = 0x00, suplib::READ_LOGICAL = 0x01, suplib::STRIP = 0x02, suplib::CLEAN = 0x04,

  suplib::RAW = 0x08 }

  Control options for getrecord.

Functions

- istream & suplib::getrecord (istream &is, string &str, int opt, char delim, char continuation)

  Reads physical and logical lines.

6.2.1 Enumeration Type Documentation

6.2.1.1 enum suplib::readopt

Control options for getrecord.

Enumerator:

READ_PHYS A single physical line makes up a record. A physical line is defined to be all characters up to the delim character. The delim character is not returned. This ignores any line continuation character. This is the default.

READ_LOGICAL A record may span multiple physical lines if the end-of-line delimiter is preceded by the continuation character. For ease of use, whitespace characters between the continuation character and the delim character are ignored.

STRIP remove all trailing whitespace from the end of a record.

CLEAN on a line with a continuation character, remove any white space following the continuation character, and add a delimiter directly following the continuation character. It has no effect on physical records or the final line in a continued logical record.

RAW return the input record (logical or physical) as is, without removing white space. end of line delimiters will be returned, but because they must be explicitly place din the string (becasue getline() is used to read from the input stream and it silently removes delimiters), there may be an extra delimiter at the end of the record if the input stream ended without a trailing delimiter

Definition at line 47 of file io.h.

6.2.2 Function Documentation

6.2.2.1 istream & suplib::getrecord (istream & *is*, string & *str*, int *opt*, char *delim*, char *continuation*)

Reads physical and logical lines.

Parameters:

is the istream from which to read.

str the string into which to read.

opt the control options.

delim the character which indicates the end of a physical line.

continuation the character directly proceeding the delim which indicates a logical record continues on the following physical line.

getrecord reads a line from the specified istream. It always reads a complete line, enlarging the string as necessary. The opt argument specifies a set of control flag, which are created by logically OR'ing suplib::readopt values.

Returns:

It returns the input istream after attempting to read lines from it.

Definition at line 60 of file getrecord.cc.

6.3 String

Functions

- bool suplib::iscomment (const string &str, const string &ignore, const string &comment)

  determine if the string is a comment.

- string & suplib::prune (string &str)

  remove leading and trailing white space from a string

- double suplib::str2d (const char *txt) throw (Exception)

  convert string to double-precision number

- float suplib::str2f (const char *txt) throw (Exception)

  convert string to floating-point number

- int suplib::str2i (const char *txt, int base) throw (Exception)

  convert string to integer number

- long suplib::str2l (const char *txt, int base) throw (Exception)

  convert string to long number

- unsigned long suplib::str2ul (const char *txt, int base) throw (Exception)

  convert string to long number

- string & suplib::trim (string &str)

  remove leading white space from a string

- template<typename Container>
  void suplib::tok (Container &container, string const &in, const char *const delimiters=" \t\n", bool skip=true)

  split a string into tokens

6.3.1 Function Documentation

6.3.1.1 bool suplib::iscomment (const string & *str*, const string & *ignore*, const string & *comment*)

determine if the string is a comment.

Parameters:

str the string upon which to operate

iscomment scans str to determine if the first character following all the ignore characters is a comment character. It compares the position of the first non-ignore character with the position of the first comment character. If they are the same and occur before the end of the string, it returns true. Otherwise it returns false.

Returns:

It returns true if line is a comment.

Definition at line 48 of file iscomment.cc.

6.3.1.2 string & suplib::prune (string & *str*)

remove leading and trailing white space from a string

Parameters:

str the string upon which to operate

prune deletes leading and trailing white space, where white space is defined as blanks, tabs, new lines, and carriage returns.

Returns:

It returns the passed reference

Definition at line 48 of file prune.cc.

6.3.1.3 double suplib::str2d (const char * *txt*) throw (Exception)

convert string to double-precision number

Parameters:

txt the string upon which to operate

str2d() converts the initial portion of the string pointed to by txt to type double representation. It throws an exception, of type Exception, if txt is not a legitimate double precision number.

Returns:

It returns the double precision number

Definition at line 44 of file str2d.cc.

6.3.1.4 float suplib::str2f (const char * *txt*) throw (Exception)

convert string to floating-point number

Parameters:

txt the string upon which to operate

str2f() converts the initial portion of the string pointed to by txt to type float representation. It throws an exception, of type Exception, if txt is not a legitimate floating point number.

Returns:

It returns the floating point number

Definition at line 39 of file str2f.cc.

6.3.1.5 int suplib::str2i (const char * *txt*, int *base*) throw (Exception)

convert string to integer number

Parameters:

txt the string upon which to operate

str2i() converts the initial portion of the string pointed to by txt to type integer representation. It throws an exception, of type Exception, if txt is not a legitimate integer.

Returns:

It returns the integer.

Definition at line 40 of file str2i.cc.

6.3.1.6 long suplib::str2l (const char * *txt*, int *base*) throw (Exception)

convert string to long number

Parameters:

txt the string upon which to operate

str2l() converts the initial portion of the string pointed to by txt to type long representation. It throws an exception, of type Exception, if txt is not a legitimate long.

Returns:

It returns the long.

Definition at line 44 of file str2l.cc.

6.3.1.7 unsigned long suplib::str2ul (const char * *txt*, int *base*) throw (Exception)

convert string to long number

Parameters:

txt the string upon which to operate

str2ul() converts the initial portion of the string pointed to by txt to type unsigned long representation. It throws an exception, of type Exception, if txt is not a legitimate long.

Returns:

It returns the long.

Definition at line 44 of file str2ul.cc.

6.3.1.8 template<typename Container> template< typename Container > void suplib::tok (Container & *container*, string const & *in*, const char *const *delimiters* = " \t\n", bool *skip* = true) [inline]

split a string into tokens

Parameters:

container the reference to a C++ container object into which the parsed string goes.

in the string to parse.

delimiters the characters which delimit tokens.

skip consecutive delimiters are skipped.

Splits the input string, in, on one or more of the characters in delimiters. The tokens are placed in container.

Returns:

void

Definition at line 60 of file str.h.

6.3.1.9 string & suplib::trim (string & *str*)

remove leading white space from a string

Parameters:

str the string upon which to operate

trim deletes leading white space, where white space is defined as blanks, tabs, new lines, and carriage returns.

Returns:

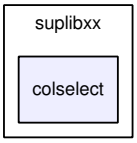
It returns the passed reference

Definition at line 49 of file trim.cc.

Chapter 7

Directory Documentation

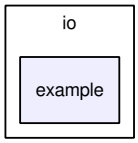
7.1 /data/macabretmp/dj/suplibxx/suplibxx/colselect/ Directory Reference



Files

- file **colselect.cc**
- file **colselect.h**
- file **match.cc**

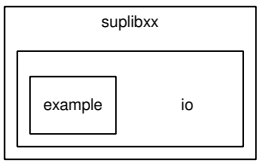
7.2 /data/macabretmp/dj/suplibxx/suplibxx/io/example/ Directory Reference



Files

- file **example_getrecord.cc**

7.3 /data/macabretmp/dj/suplibxx/suplibxx/io/ Directory Reference



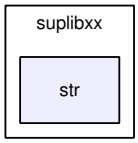
Directories

- directory example

Files

- file **getrecord.cc**
- file **io.h**

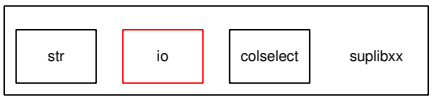
7.4 /data/macabretmp/dj/suplibxx/suplibxx/str/ Directory Reference



Files

- file **iscomment.cc**
- file **prune.cc**
- file **str.h**
- file **str2d.cc**
- file **str2f.cc**
- file **str2i.cc**
- file **str2l.cc**
- file **str2ul.cc**
- file **trim.cc**

7.5 /data/macabretmp/dj/suplibxx/suplibxx/ Directory Reference



Directories

- directory colselect
- directory io
- directory str

Chapter 8

Namespace Documentation

8.1 suplib Namespace Reference

The suplib namespace encompasses all of the functions in the suplib++ library.

Enumerations

- enum readopt {

  READ_PHYS = 0x00, READ_LOGICAL = 0x01, STRIP = 0x02, CLEAN = 0x04,

  RAW = 0x08 }

  Control options for getrecord.

Functions

- void colselect (const vector< string > &icolumns, const vector< string > &exact_add, const vector< string > &regex_add, const vector< string > &exact_del, const vector< string > &regex_del, vector< string > &ocolumns) throw (Exception)

  select columns based on exact/regex matching/exclusion.

- bool match (const string &str, const string &pattern) throw (Exception)

  handles Perl regular expression matching.

- istream & getrecord (istream &is, string &str, int opt, char delim, char continuation)

Reads physical and logical lines.

- bool iscomment (const string &str, const string &ignore, const string &comment)

  determine if the string is a comment.

- string & prune (string &str)

  remove leading and trailing white space from a string

- string & trim (string &str)

  remove leading white space from a string

- float str2f (const char *txt) throw (Exception)

  convert string to floating-point number

- double str2d (const char *txt) throw (Exception)

  convert string to double-precision number

- int str2i (const char *txt, int base) throw (Exception)

  convert string to integer number

- long str2l (const char *txt, int base) throw (Exception)

  convert string to long number

- unsigned long str2ul (const char *txt, int base) throw (Exception)

  convert string to long number

- template<typename Container>
  void tok (Container &container, string const &in, const char *const delimiters=" \t\n", bool skip=true)

  split a string into tokens

8.1.1 Detailed Description

The suplib namespace encompasses all of the functions in the suplib++ library.

Index