

suplibxx  
1.3.13

Generated by Doxygen 1.8.15



<b>1 The C++ suplib Library</b>	<b>1</b>
1.1 Copyright	1
1.2 Column Name Selection	1
1.3 Input/Output	1
1.4 String	1
<b>2 I/O Examples</b>	<b>3</b>
<b>3 Module Index</b>	<b>5</b>
3.1 Modules	5
<b>4 Namespace Index</b>	<b>7</b>
4.1 Namespace List	7
<b>5 Module Documentation</b>	<b>9</b>
5.1 Column Selection	9
5.1.1 Detailed Description	9
5.1.2 Function Documentation	9
5.1.2.1 colselect()	9
5.1.2.2 match()	10
5.2 Input/Output	11
5.2.1 Detailed Description	11
5.2.2 Enumeration Type Documentation	11
5.2.2.1 readopt	11
5.2.3 Function Documentation	12
5.2.3.1 getrecord()	12
5.3 String	13
5.3.1 Detailed Description	13
5.3.2 Function Documentation	13
5.3.2.1 iscomment()	13
5.3.2.2 prune()	14
5.3.2.3 str2d()	14
5.3.2.4 str2f()	15
5.3.2.5 str2i()	15
5.3.2.6 str2l()	16
5.3.2.7 str2ul()	16
5.3.2.8 trim()	17
<b>6 Namespace Documentation</b>	<b>19</b>
6.1 suplib Namespace Reference	19
6.1.1 Detailed Description	20



# 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

Copyright (C) 2006 Smithsonian Astrophysical Observatory

This file is part of suplibxx

suplibxx is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

suplibxx is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor Boston, MA 02110-1301, USA

### 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 strsr;
    strsr << " now, is, the, time ";
    expected = " now, is, the, time ";
    suplib::getrecord( strsr, 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 strsr;
    strsr << " now, is, the, time \n";
    expected = " now, is, the, time";
    suplib::getrecord( strsr, 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 strsr;
    strsr << " 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( strsr, 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 strsr;
    strsr << " 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( strsr, 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 strsr;
    strsr << " now, is, the, time \n";
    expected = " now, is, the, time";
    suplib::getrecord( strsr, 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 strsr;
    strsr << " 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( strsr, 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 strsr;
    strsr << " 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( strsr, 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 strsr;
    strsr << " 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( strsr, 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:

Column Selection . . . . .	9
Input/Output . . . . .	11
String . . . . .	13



## Chapter 4

# Namespace Index

### 4.1 Namespace List

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

[suplib](#)

The suplib namespace encompasses all of the functions in the suplib++ library . . . . . [19](#)



## Chapter 5

# Module Documentation

### 5.1 Column Selection

#### Functions

- void [suplib::colselect](#) (const std::vector< std::string > &icolumns, const std::vector< std::string > &exact\_add, const std::vector< std::string > &regex\_add, const std::vector< std::string > &exact\_del, const std::vector< std::string > &regex\_del, std::vector< std::string > &ocolumns)  
*select columns based on exact/regex matching/exclusion.*
- bool [suplib::match](#) (const std::string &str, const std::string &pattern)  
*handles Perl regular expression matching.*

#### 5.1.1 Detailed Description

#### 5.1.2 Function Documentation

##### 5.1.2.1 colselect()

```
void suplib::colselect (
    const std::vector< std::string > & icolumns,
    const std::vector< std::string > & exact_add,
    const std::vector< std::string > & regex_add,
    const std::vector< std::string > & exact_del,
    const std::vector< std::string > & regex_del,
    std::vector< std::string > & ocolumns )
```

select columns based on exact/regex matching/exclusion.

**Parameters**

<i>icolumns</i>	vector of column names on which to operate
<i>exact_add</i>	column names to add to output
<i>regex_add</i>	regular expressions used to add to output
<i>exact_del</i>	column names to exclude from output
<i>regex_del</i>	regular expressions used to exclude from output
<i>ocolumns</i>	set of column names which were selected

**Exceptions**

<i>Exception</i>	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 57 of file colselect.cc.

**5.1.2.2 match()**

```
bool suplib::match (
    const std::string & str,
    const std::string & pattern )
```

handles Perl regular expression matching.

**Parameters**

<i>str</i>	the string
<i>pattern</i>	the regular expression pattern with which to compare

**Exceptions**

<i>Exception</i>	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 50 of file match.cc.

## 5.2 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

- `std::istream & suplib::getrecord` (`std::istream &is`, `std::string &str`, `int opt=READ_PHYS`, `char delim='\n'`, `char continuation='\'`)

*Reads physical and logical lines.*

#### 5.2.1 Detailed Description

#### 5.2.2 Enumeration Type Documentation

##### 5.2.2.1 readopt

```
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 41 of file io.h.

## 5.2.3 Function Documentation

### 5.2.3.1 getrecord()

```
std::istream & suplib::getrecord (
    std::istream & is,
    std::string & str,
    int opt = READ_PHYS,
    char delim = '\n',
    char continuation = '\\' )
```

Reads physical and logical lines.

#### Parameters

<i>is</i>	the istream from which to read.
<i>str</i>	the string into which to read.
<i>opt</i>	the control options.
<i>delim</i>	the character which indicates the end of a physical line.
<i>continuation</i>	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 53 of file getrecord.cc.



## 5.3 String

### Functions

- bool `suplib::iscomment` (const std::string &str, const std::string &ignore="\t", const std::string &comment="#")  
*determine if the string is a comment.*
- std::string & `suplib::prune` (std::string &str)  
*remove leading and trailing white space from a string*
- double `suplib::str2d` (const char \*txt)  
*convert string to double-precision number*
- float `suplib::str2f` (const char \*txt)  
*convert string to floating-point number*
- int `suplib::str2i` (const char \*txt, int base=10)  
*convert string to integer number*
- long `suplib::str2l` (const char \*txt, int base=10)  
*convert string to long number*
- unsigned long `suplib::str2ul` (const char \*txt, int base=10)  
*convert string to long number*
- std::string & `suplib::trim` (std::string &str)  
*remove leading white space from a string*

#### 5.3.1 Detailed Description

#### 5.3.2 Function Documentation

##### 5.3.2.1 iscomment()

```
bool suplib::iscomment (
    const std::string & str,
    const std::string & ignore = " \t",
    const std::string & comment = "#" )
```

determine if the string is a comment.

#### Parameters

<i>str</i>	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 45 of file iscomment.cc.

**5.3.2.2 prune()**

```
std::string & suplib::prune (  
    std::string & str )
```

remove leading and trailing white space from a string

**Parameters**

<i>str</i>	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 46 of file prune.cc.

**5.3.2.3 str2d()**

```
double suplib::str2d (  
    const char * txt )
```

convert string to double-precision number

**Parameters**

<i>txt</i>	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.

**5.3.2.4 str2f()**

```
float suplib::str2f (
    const char * txt )
```

convert string to floating-point number

**Parameters**

<i>txt</i>	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 41 of file str2f.cc.

**5.3.2.5 str2i()**

```
int suplib::str2i (
    const char * txt,
    int base = 10 )
```

convert string to integer number

**Parameters**

<i>txt</i>	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 43 of file str2i.cc.

**5.3.2.6 str2l()**

```
long suplib::str2l (  
    const char * txt,  
    int base = 10 )
```

convert string to long number

**Parameters**

<i>txt</i>	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 46 of file str2l.cc.

**5.3.2.7 str2ul()**

```
unsigned long suplib::str2ul (  
    const char * txt,  
    int base = 10 )
```

convert string to long number

**Parameters**

<i>txt</i>	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 46 of file str2ul.cc.

**5.3.2.8 trim()**

```
std::string & suplib::trim (  
    std::string & str )
```

remove leading white space from a string

**Parameters**

<i>str</i>	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 48 of file trim.cc.



## Chapter 6

# Namespace Documentation

### 6.1 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 std::vector< std::string > &icolumns, const std::vector< std::string > &exact\_add, const std::vector< std::string > &regex\_add, const std::vector< std::string > &exact\_del, const std::vector< std::string > &regex\_del, std::vector< std::string > &ocolumns)  
*select columns based on exact/regex matching/exclusion.*
- bool `match` (const std::string &str, const std::string &pattern)  
*handles Perl regular expression matching.*
- std::istream & `getrecord` (std::istream &is, std::string &str, int opt=`READ_PHYS`, char delim='\n', char continuation='\')
- std::string & `trim` (std::string &str)  
*remove leading white space from a string*
- std::string & `prune` (std::string &str)  
*remove leading and trailing white space from a string*
- bool `iscomment` (const std::string &str, const std::string &ignore=" \t", const std::string &comment="#")  
*determine if the string is a comment.*
- float `str2f` (const char \*txt)  
*convert string to floating-point number*

- double `str2d` (const char \*txt)  
*convert string to double-precision number*
- int `str2i` (const char \*txt, int base=10)  
*convert string to integer number*
- long `str2l` (const char \*txt, int base=10)  
*convert string to long number*
- unsigned long `str2ul` (const char \*txt, int base=10)  
*convert string to long number*

### 6.1.1 Detailed Description

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



# Index

## CLEAN

Input/Output, [11](#)

## colselect

Column Selection, [9](#)

## Column Selection, [9](#)

colselect, [9](#)

match, [10](#)

## getrecord

Input/Output, [12](#)

## Input/Output, [11](#)

CLEAN, [11](#)

getrecord, [12](#)

RAW, [11](#)

READ\_LOGICAL, [11](#)

READ\_PHYS, [11](#)

readopt, [11](#)

STRIP, [11](#)

## iscomment

String, [13](#)

## match

Column Selection, [10](#)

## prune

String, [14](#)

## RAW

Input/Output, [11](#)

## READ\_LOGICAL

Input/Output, [11](#)

## READ\_PHYS

Input/Output, [11](#)

## readopt

Input/Output, [11](#)

## str2d

String, [14](#)

## str2f

String, [15](#)

## str2i

String, [15](#)

## str2l

String, [16](#)

## str2ul

String, [16](#)

## String, [13](#)

iscomment, [13](#)

prune, [14](#)

str2d, [14](#)

str2f, [15](#)

str2i, [15](#)

str2l, [16](#)

str2ul, [16](#)

trim, [17](#)

## STRIP

Input/Output, [11](#)

## suplib, [19](#)

## trim

String, [17](#)